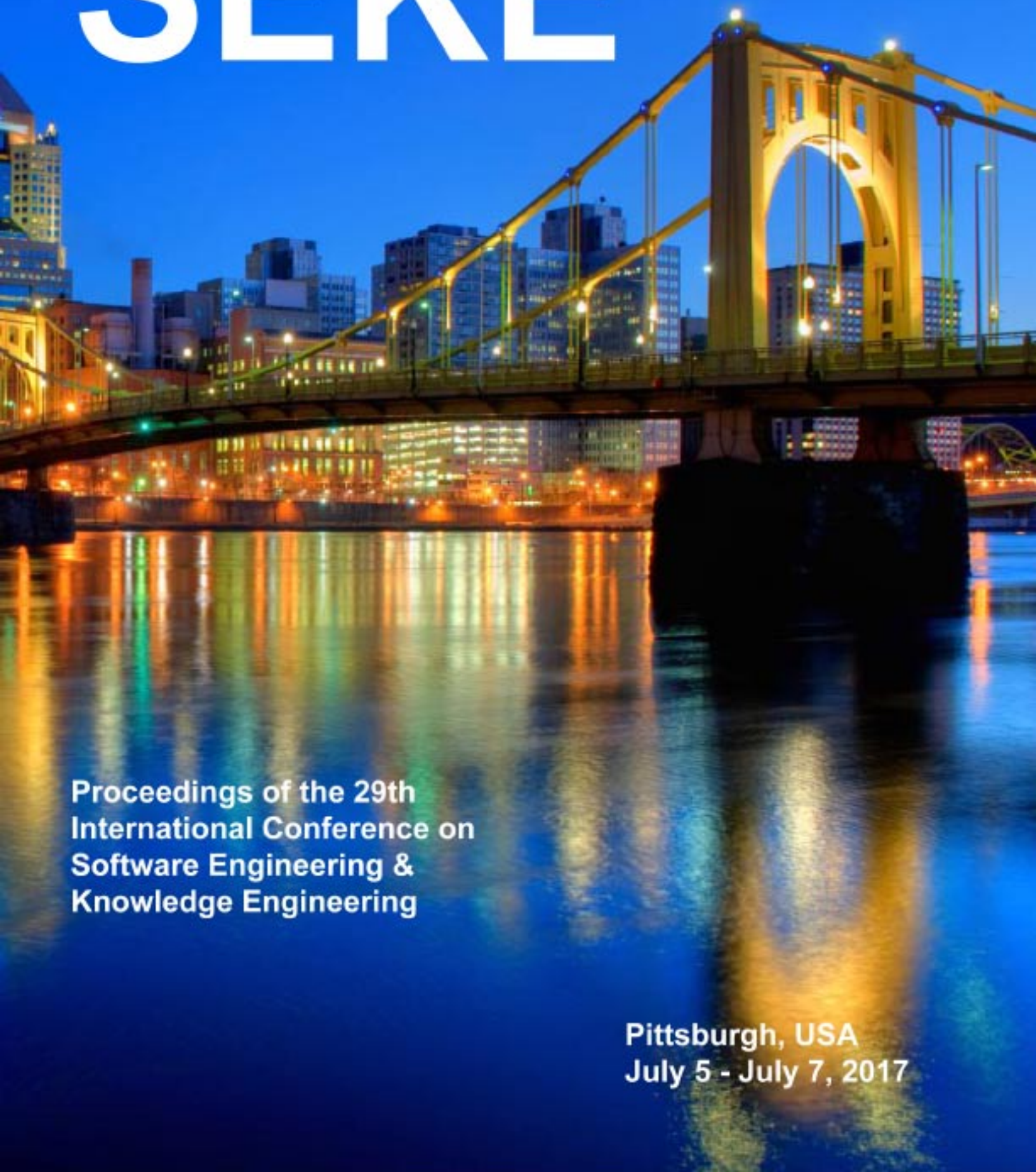
SEKE 2017



Proceedings of the 29th International Conference on Software Engineering & Knowledge Engineering

Pittsburgh, USA
July 5 - July 7, 2017

PROCEEDINGS

SEKE 2017

The 29th International Conference on Software Engineering & Knowledge Engineering

Sponsored by

KSI Research Inc. and Knowledge Systems Institute, USA

Technical Program

July 5 – 7, 2017

Wyndham Pittsburgh University Center, Pittsburgh, USA

Organized by

KSI Research Inc. and Knowledge Systems Institute, USA

ISBN: 1-891706-41-1

ISSN: 2325-9000 (print)

2325-9086 (online)

DOI reference number: 10.18293/SEKE2017

Publisher Information:

KSI Research Inc. and Knowledge Systems Institute

156 Park Square

Pittsburgh, PA 15238 USA

Tel: +1-412-606-5022

Fax: +1-847-679-3166

Email: seke@ksiresearch.org

Web: http://ksiresearchorg.ipage.com/seke/seke17.html

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

Printed by KSI Research Inc. and Knowledge Systems Institute

FOREWORD

Welcome to the 29th International Conference on Software Engineering and Knowledge Engineering (SEKE), in Wyndham Pittsburgh University Center, Pittsburgh, USA. In last 29 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 2017 and its rich assortment of activities, but also in enjoying the beautiful Pittsburgh Area.

This year, we received 195 submissions from 34 countries. Through a rigorous review process where almost all of the submitted papers received at least three reviews, and only a few with two reviews, we were able to select 69 full papers for the general conference (35.3 percent), and 61 short papers (31.5 percent). Out of that, 9 papers have been accepted for the DBKE workshop, 12 papers have been accepted for the DISA workshop, and 4 papers have been accepted for the CISS workshop. 130 papers are scheduled for presentation in thirty five sessions during the conference. In addition, the technical program includes two excellent keynote speeches and a plenary talk from the co-locating DMSVLSS2017 conference, 9 posters and demo presentations, as well as workshops on Big Data Research and Development in Knowledge Engineering (BDKE), Data Intensive Services based Application (DISA), and Conceptual Integrity of Software Systems (CISS).

The high quality of the SEKE 2017 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 my 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 my 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 2017. Our heartfelt appreciation goes to Dr. Jerry Gao, San Jose State University, USA, the Conference Chair, for his help and experience, and to the Program Committee Co-Chairs, Dr. Oscar Pereira, University of Aveiro, Portugal, Angelo Perkusich, Federal, University of Campina Grande, Brazil, Huiqun Yu, East China University of Science and Technology, China for their outstanding team work. In addition, we am truly grateful to the workshop organizers Dr. Zheng Xu, Tsinghua University, China, Dr. Honghao Gao, Shanghai University, China, and Prof. Iaakov Exman, The Jerusalem College of Engineering, Israel for their great job in organizing Workshops on Big Data Research and Development in Knowledge Engineering, Data Intensive Services based Application, and Conceptual Integrity of Software Systems respectively.

Moreover, we would like to express my great appreciation to all of the conference organization committee members, including the Publicity Chair, Dr. Robert Heinrich, Karlsruhe Institute of Technology, Germany, Demo & Poster Session Chair, Dr. Masoud Sadjadi, Florida International University, USA. Moreover, we would like to appreciate and recognize our Conference Liaisons in different regions for their important contributions: Asia Liaison – Hironori Washizaki, Waseda University, Japan; Australia Liaison – Aneesh Krishna, Curtin University of Technology, Australia; Europe Liaison – Raul Garcia Castro, Universidad Politecnica de Madrid, Spain; India Liaison – Swapan Bhattacharya, National Institute of Technology Karnataka, Surathakl, India; and South America Liaison - Jose Carlos Maldonado, ICMC-USP, Brazil.

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 Pittsburgh to participate in SEKE 2017.

Xudong He, Florida International University, USA, Program Committee Chair
Oscar Pereira, University of Aveiro, Portugal , Program Committee Co-Chair
Angelo Perkusich, Federal University of Campina Grande, Brazil, Program Committee Co-Chair
Huiqun Yu, East China University of Science and Technology, China, Program Committee Co-Chair

SEKE 2017

The 29th International Conference on Software Engineering & Knowledge Engineering

July 5– 7, 2017

Wyndham Pittsburgh University Center, Pittsburgh, USA

Conference Organization

CONFERENCE CHAIR

Jerry Gao, San Jose State University, USA

PROGRAM COMMITTEE CHAIR

Xudong He, Florida International University, USA

PROGRAM COMMITTEE CO-CHAIRS

Oscar Pereira, University of Aveiro, Portugal
Angelo Perkusich, Federal University of Campina Grande, Brazil
Huiqun Yu, East China University of Science and Technology, China

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
Izzat Alsmadi, Boise State 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
Kyungmin Bae, Pohang University of Science and Technology, Korea
Ebrahim Bagheri, Ryerson University, Canada
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
Swapan Bhattacharya, NITK, Surathakl, India
Alessandro Bianchi, University of Bari, Italy
Ivo Bukovsky, Czech Technical University in Prague, Czech Republic
Jiannong Cao, Hong Kong Polytechnic University, China
Raul Garcia Castro, Universidad Politecnica de Madrid, Spain
Nabendu Chaki, University of Calcutta, India
Chih-Hung Chang, Hsiuping University of Science and Technology, Taiwan
Keith Chan, Hong Kong Polytechnic University, Hong Kong
Kuang-nan Chang, Eastern Kentucky University, USA
Lily Chang, University of Wisconsin, Platteville, USA
Meiru Che, University of Texas at Austin, USA
Wen-Hui Chen, National Taipei University of Technology, Taiwan
William Chu, Tunghai University, Taiwan
Stelvio Cimato, University of Milan, Italy
Nelly Condori-fernandez, VU University Amsterdam, The Netherlands
Fabio M. Costa, Universidade Federal de Goias, Brazil
Maria Francesca Costabile, University of Bari, Italy
Jose Luis De La Vara, Carlos III University of Madrid, Spain
Massimiliano Di Penta, University of Sannio, Italy
Scott Dick, University of Alberta, Canada
Junhua Ding, East Carolina University, USA
Fei Dong, Google Inc., USA
Zhijian Dong, Middle Tennessee State University, USA
Derek Doran, Wright 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 Ebnenasir, Michigan Technological University, USA
Raimund Ege, NIU, USA
Magdalini Eirinaki, San Jose State University, USA
Mahmoud Elish, King Fahd University of Petroleum and Minerals, Saudi Arabia
Ruby ElKharboutly, Quinnipiac University, Canada
Davide Falessi, Cal Poly, USA
Behrouz Far, University of Calgary, Canada
Liana Fong, IBM, USA
Ellen Francine Barbosa, University of Sao Paulo, Brazil
Fulvio Frati, University of Milan, Italy
Yujian Fu, Alabama A & M University, USA
Jerry Gao, San Jose State University, USA
Kehan Gao, Eastern Connecticut State University, USA
Felix Garcia, University of Castilla-La Mancha, Spain

Olivier Le Goaer, 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
Hassan Haghighi, Shahid Beheshti University, Iran
Hao Han, National Institute of Informatics, Japan
Xudong He, Florida International University, USA
Robert Heinrich, Karlsruhe Institute of Technology, Germany
Miguel Herranz, University of Alcala, Spain
Rubing Huang, Jiangsu University, China
Shihong Huang, Florida Atlantic University, USA
Hamdy Ibrahim, University of Calgary, Canada
Bassey Isong, North-West University, South Africa
Clinton Jeffery, University of Idaho, USA
Jason Jung, Chung-Ang University, South Korea
Selim Kalayci, Florida International University, USA
Marcos Kalinowski, Fluminense Federal University, Brazil
Pankaj Kamthan, Concordia University, Canada
Ananya Kanjilal, B.P. Poddar Institute of Technology and Management, India
Eric Kasten, Michigan State University, USA
Taghi Khoshgoftaar, Florida Atlantic University, USA
Claudiu V. Kifor, Lucian Blaga University of Sibiu, Romania
Jun Kong, North Dakota State University, USA
Aneesh Krishna, Curtin University of Technology, Australia
Vinay Kulkarni, Tata Consultancy Services, India
Jeff Lei, University of Texas at Arlington, USA
Meira Levy, Shenkar College of Engineering and Design, Israel
Bixin Li, Southeast University, China
Xin Li, Google Inc., USA
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
Xiaoqing Frank Liu, Missouri University of Science and Technology, USA
KaiKai Liu, San Jose State University, USA
Shih-hsi Liu, California State University, Fresno, USA
Ting Liu, Xian Jiaotong University, China
Xiaodong Liu, Edinburgh Napier University, United Kingdom
Yi Liu, Georgia College & State University, USA
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 Uberlendia, Brazil
Beatriz Marin, Universidad Diego Portales, Chile
Riccardo Martoglia, University of Modena and Reggio Emilia, Italy
Santiago Matalonga, Universidad ORT Uruguay, Uruguay
Hong Mei, Peking University, China
Hsing Mei, Fu Jen Catholic Unicersity, Taiwan
Andre Menolli, Universidade Estadual do Norte do Parana (UENP), Brazil
Ali Mili, NJIT, USA
Alok Mishra, Atilim University, Turkey
Manuel Mora, Autonomous University of Aguascalientes, Mexico
Hiroyuki Nakagawa, Osaka University, Japan

Kia Ng, ICSRiM - University of Leeds, United Kingdom
Allen Nikora, Jet Propulsion Laboratory, USA
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
Daniel Plante, Stetson University, USA
Rick Rabiser, Johannes Kepler University, Austria
Filip Radulovic, Universidad Politecnica de Madrid, Spain
Claudia Raibulet, University of Milan, Italy
Damith C. Rajapakse, National University of Singapore, Singapore
Rajeev Raje, IUPUI, USA
Santanu Ku Rath, National Institute of Technology, India
Henrique Rebelo, Universidade Federal de Pernambuco, Brazil
Marek Reformat, University of Alberta, Canada
Robert Reynolds, Wayne State University, USA
Hassan Reza, University of North Dakota, USA
Elder Macedo Rodrigues, Pontifical Catholic University of Rio Grande do Sul, Brazil
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
Andreas Schoenberger, Siemens AG, Germany
Michael Shin, Texas Tech University, USA
Martin Solari, Universidad ORT Uruguay, Uruguay
Qinbao Song, Xi'an Jiaotong University, China
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
Sarkar Tanmoy, Microsoft Corporation, USA
Chuanqi Tao, Nanjing University of Science and Technology, China
Jeff Tian, Southern Methodist University, USA
Mark Trakhtenbrot, Holon Institute of Technology, Israel
Peter Troeger, TU Chemnitz, Germany
T. H. Tse, The University of Hong Kong, Hong Kong
Christelle Urtado, LGI2P Ecole des Mines d'Ales, France
Sylvain Vauttier, Ecole des mines d'Ales, France
Silvia Vergilio, Federal University of Parana (UFPR), Brazil
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
Jiacun Wang, Monmouth University, USA
Xiaoyin Wang, University of Texas at San Antonio, USA
Ye Wang, Zhejiang Gongshang University, China
Yong Wang, New Mexico Highlands University, USA
Young Wang, Chongqing University of Posts and Telecommunications, China
Zhongjie Wang, Harbin Institute of Technology, China
Ziyuan Wang, Nanjing University of Posts and Telecommunications, China
Hironori Washizaki, Waseda University, Japan

Victor Winter, University of Nebraska at Omaha, 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, Bath Spa University, United Kingdom
Huiqun Yu, East China University of Science and Technology, China
Jiang Yue, Fujian Normal University, 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
Jiang Zheng, ABB US Corporate Research Center, USA
Zhigao Zheng, Central China Normal University, USA
Hong Zhu, Oxford Brookes University, UK
Huibiao Zhu, East China Normal University, China
Xingquan Zhu, Florida Atlantic University, USA
Eugenio Zimeo, University of Sannio, Italy

PUBLICITY CHAIR

Robert Heinrich, Karlsruhe Institute of Technology, Germany

ASIA LIAISON

Hironori Washizaki, Waseda University, Japan

AUSTRALIAN LIAISON

Aneesh Krishna, Curtin University of Technology, Australia

EUROPE LIAISON

Raul Garcia Castro, Universidad Politecnica de Madrid, Spain

INDIA LIAISON

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

SOUTH AMERICA LIAISON

Jose Carlos Maldonado, ICMC-USP, Brazil

Keynote

Actor-Oriented Programming for the Internet of Things

Professor Gul Agha
Department of Computer Science
Department of Electrical and Computer Engineering
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign

Abstract

The Internet of Things involves networked embedded applications in an open distributed system. Critical requirements of IoT applications include scalability, robustness, real-time responsiveness, energy efficiency and adaptive control. These characteristics are common in biological systems--autonomous agents who symbiotically use model-based data acquisition and feeback control. Individual agents have partial information and coordinate in real-time to create a robust system. The Actor model has been used to build scalable software systems such as Twitter, Linkedin, Facebook Chat. The Actor model can be extended with probabilistic execution, continuous variables, control and coordination abstractions to facilitate building IoT applications. I will describe these extensions and then discuss how IoT systems with large numbers of actors can be tested and verified for parallel performance and energy behavior using techniques such as Runtime Verification, Statistical Model Checking and Euclidean Modeling Checking.

About the Speaker

Gul Agha is Professor of Computer Science at the University of Illinois at Urbana-Champaign and co-founder of Embedor Technologies, a start-up providing solutions for civil infrastructure monitoring for smart cities. Dr. Agha is best known for his work on the Actor model: his thesis on actors, published by MIT Press, is the most widely cited work on the topic. Dr. Agha has developed a number of novel techniques in the areas of Statistical Model Checking, Euclidean Model Checking, an Energy complexity model for parallel algorithms, software testing and mechanisms for multi-actor coordination. Dr. Agha is a Fellow of the IEEE. He previously served as Editor-in-Chief of IEEE Parallel and Distributed Technology (1994-98) and ACM Computing Surveys (1999-2007), and is currently the Editor-in-Chief of IEEE Computing Now.

Keynote

User-centric Techniques for Big Data Exploration

Professor Panos Chrysanthis
Department of Computer Science
University of Pittsburgh

Abstract

As the amount of data being generated every day increases exponentially, the term "Bid Data" has been adopted to represent the challenge of large-scale data processing. Given the volume of data, the challenge is how to avoid overwhelming the users with irrelevant results. In this talk, we will focus on recent research challenges and opportunities in the emerging area of Data Exploration that aim to guide users to reveal valuable insights from large volumes of data (e.g., financial and scientific databases). We will overview the recent data analysis techniques underlying some of the new data exploration tools, and we will introduce our new framework, called Preferential Diversity (PrefDiv), which is capable of generating results that are not only relevant to users' preference but are also diverse. We will also briefly present our annotation framework, designed for biological data, that enables users to collaboratively explore large data sets.

About the Speaker

Panos K. Chrysanthis is a Professor of Computer Science and a founder and director of the Advanced Data Management Technologies Laboratory at the University of Pittsburgh. He is also an adjunct Professor at Carnegie-Mellon University. His research interests lie within the areas of data management and has fostered interdisciplinary collaborations between computer science, medicine, astronomy and mechanical engineering, both within and outside the University of Pittsburgh. His research contributions in principles, algorithms and prototypes to data management have been documented in more than 150 papers in top journals and prestigious, peer-reviewed conferences and workshops. His editorial service includes VLDB J (2001-2007), IEEE TKDE (2010-present) and DAPD (2010-present). Chrysanthis is a NSF CAREER award recipient, an ACM Distinguished Scientist and a Senior Member of IEEE. He was honored with seven teaching awards and in 2015, he received the University of Pittsburgh's Provost Award for Excellence in Mentoring (doctoral students).

Table of Contents

Data Intensive Services based Application-II

Software Process

Literature Review

Big Data R&D In Knowledge Engineering-I

Performance

Formal Methods

Feature Selection

Requirements Modeling

Quality Assurance I

Search

Data Mining II

Quality Assurance II

Recommendation

Big Data R&D In Knowledge Engineering-II

Conceptual Integrity of Software Systems

Code Analysis I

Agent

Software Education

Ontology

Development Tools

Data Analysis

Posters and Demos

Note:
(S) indicates a short paper.
(P) indicates a poster.

Answering Who/When, What, How, Why through Constructing Data Graph, Information Graph, Knowledge Graph and Wisdom Graph

Lixu Shao*, Yucong Duan**†, Xiaobing Sun‡, Honghao Gao§, Donghai Zhu¶ and Weikai Miao‖
*College of Information Science and Technology,
State Key Laboratory of Marine Resource Utilization in the South China Sea,Hainan University, China
Email: 751486692,841213174,466971642@qq.com
‡School of Information Engineering Yangzhou University, China
Email: xbsun@yzu.edu.cn
§Computing Center, Shanghai University, Shanghai, China
Email:gaohonghao@shu.edu.cn
‖Software Engineering Institute, East China Normal University, China
Email:wkmiao@sei.ecnu.edu.cn

***Abstract*—Knowledge graphs have been widely adopted, in large part owing to their schema-less nature. It enables knowledge graphs to grow seamlessly and allows for new relationships and entities as needed. Natural language questions are the most intuitive way of formulating an information need. People can formulate questions to express their information needs. Natural language questions as a query language present an ideal compromise between keyword and structured querying. Questions can be used to express complex information needs that cannot be expressed as keywords without a significant loss in structure and semantics. Knowledge graph has abundant natural semantics and can contain various and more complete information. Its expression mechanism is closer to natural language. We propose to clarify the expression of knowledge graph as a whole. We use knowledge graph to solve the Five Ws problems respectively which are guided by interrogative words such as who/when, what, how and why. We also propose to specify knowledge graph in a progressive manner as four basic forms including data graph, information graph, knowledge graph and wisdom graph.**

1. Introduction

In terms of usability, questions are a more accessible medium for expressing complex information needs. The "What, How, Who, When, Why" (Five Ws) are questions whose answers are considered to be the basis of information-gathering and problem-solving. Traditionally, users interact with search engines by providing keywords to the search engine and obtaining a list of documents that best match those keywords. The two major drawbacks of this pattern are the answer to granularity and query expression [1]. Keyword queries are highly "telegraphic" [2] which means that they have limited expressiveness with missing verbs, prepositions, clauses and phrase clues. With this limitation, users cannot express complex information needs. Knowledge graph has become a powerful tool to represent knowledge in the form of a labelled directed graph and to give semantics to textual information. A knowledge graph is a graph constructed by representing each item, entity and user as nodes, and linking those nodes that interact with each other via edges. However, there is still a lack of a standard definition of knowledge graph. In [10] the authors elaborated conceptual approaches for defining data, information, and knowledge. Data is acquired by observing the basic individual items of numbers or other information, but on their own, without context, they have no information. Information is conveyed through the context of data and data combinations, and may be suitable for analysis and interpretation. Knowledge is the general understanding and consciousness gained from the accumulated information, and the experience is adjusted so that a new background can be envisaged. Wisdom is an extrapolative and non-deterministic, non-probabilistic process. It calls upon all the previous levels of consciousness, and specifically upon special types of human programming [12]. We propose to clarify the expression of knowledge graph as a whole. We use knowledge graph to solve the Five Ws problems respectively which are instructed by interrogative words such as who/when, what, how and why. Each of them can be widely used to explore and evaluate a variety of knowledge theories and systems. We show the progressive forms of knowledge type in Table 1. Correspondingly, we propose to specify knowledge graph in a progressive manner as four basic forms including data graph, information graph, knowledge graph and wisdom graph.

In the rest of this paper, we elaborate how to use knowledge graph to solve the problem guided by interrogative words including "who & when" "what" "how" "why" in Section 2, 3, 4, and 5 respectively. The related works are elaborated in Section 6. And we give our conclusions in Section 7.

DOI Reference Number 10.18293/SEKE2017-079

TABLE 1. THE PROGRESSIVE FORM OF KNOWLEDGE TYPE

(rightside: general forms)	data	information	knowledge	wisdom
Semantic load	not specified for stakeholders/machine	settled for stakeholders/machine	abstracted on known information	Known to unknown
format	discrete elements	related elements	Probabilistic or categorization	(frame or stylish expression)
knowledge answer	who/when/where	what	how	why
usage	transmission	communication	reasoning	prediction
Sub graphs	data graph	information graph	knowledge graph	wisdom pool

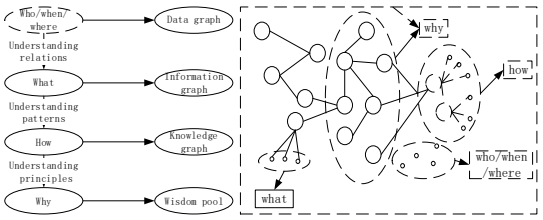


Figure 1. Relations between "Who/when/where", "What", "How" and "Why".

2. Relationship between "who/when/where", 'what", "how" and "why"

In Fig. 1 we show the relationship between "Who/When/Where", "What" "How" and "Why". We use discrete points to represent data. Data represents a fact or statement of event without relation to other things and we can use data to give answers to the questions guided by "Who/When/Where". Data requires interpretation to become information. To translate data to information, several factors must be considered. The factors involved are determined by the creator of data and the desired information. Information is data that has been given meaning by way of relational connection. This "meaning" can be useful, but does not have to be. By organizing the associated discrete data points together, we can provide answers to "what" questions. Knowledge is the appropriate collection of information and it is considered to be of great use. Knowledge is a deterministic process and can be tacit as well as explicit. When people "memorize" information, they have amassed knowledge. This knowledge has useful meaning to them, but it does not provide for, in and of itself, an integration such as would infer further knowledge. We use knowledge to answer "how" questions. The first three categories relate to the past and they deal with what has been or what is known. Only the fourth category, wisdom, deals with the future because it incorporates vision and design. With wisdom, people can create the future rather than just grasp the present and past [16]. But achieving wisdom isn't easy. People must move successively through the other three categories. By assessing and understanding the acquired knowledge, we infer what we did not know before. Wisdom can provide answers to "Why" questions.

3. Using Knowledge Graph to Answer the Question of "Who& When"

We represent entities with the logical predicate Ent(E). We represent labels with the logical predicate Lbl(L). Then an entity having labels can be defined as Ent(E) = Lbl(L). Relations are represented with the logical predicate Rel(E1,E2,R) where the relation R holds between the entities E1 and E2, for instance, R(E1,E2). We define a knowledge graph G as a set of triples of the form (s,r,t) where s,t $\in$ E and r $\in$ R. When a user presents his/her information needs by asking questions, we propose to segment questions and recognize the basic entity E1, relationship type R, and type of target entity E2. The query seeks target entity E2, where R (E1, E2) holds and mentions of entities have been linked to the corresponding knowledge graph. In general, we can present a user's question to a triad form. A triple pattern query, or simply a query is a set of triple patterns Q = $q_1, ..., q_n$ and a projection set P(Q) of variables. We require the join graph of Q, where $q_i s$ are vertices and an edge exists between every pair of vertices sharing a common variable, to be a connected graph (to avoid computing Cartesian products). P (Q) is a subset of the variables in Q, defining the output structure, typically tuples of entities. We also refer to the triple pattern set Q as a query when the projection set is not relevant for the discussion. We adapt one of the figure examples from [1] to explain. When a user enters "who is the spouse of Robert Rossellini", it can be expressed as (?x, spouse, RobertRessellini). The corresponding triple pattern query is

SELECT ?x WHERE ?x spouse Robert Rossellini.

According to the knowledge graph shown in Fig. 2 we can learn that the answer to the question is P (Q) = Ingrid Bergman. The denition of an answer to a query is a natural extension of an answer to a single triple pattern.The method of dealing with "when" questions is similar to that of "who" questions.

4. Using Knowledge Graph to Answer the Question of "What"

4.1. Inferring the property of an entity according to rules

Compared with existing modelling tools such as UML Class Diagrams, knowledge graph has abundant natural semantics, and its expression mechanism is closer to natural

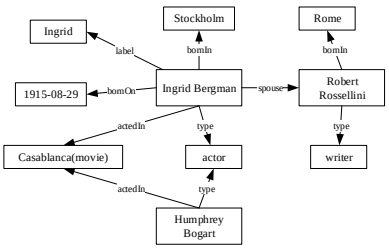


Figure 2. An example knowledge graph of people (only a fragment is shown).

language. Knowledge graph can contain various and more complete information. According to algorithm 1, we can generate rules from a large number of training data sets. When a user searches for an entity, if collection of the entity′s (relation, label) value pair matches to a certain rule, then we can know which class the entity belong to. For example, we give a set of rules about verifying vertebrate categories as shown below:

r1: (is a, flying animal) ∧ (has, feather) ∧ (is a, constant temperature animal) → bird.

r2: (is a, aquatic animal) ∧ (has, squama) ∧ (breath with, gills) → fish.

r3: (is a, variable temperature animal) ∧ (has, squama) ∧ (breath with, lung) → reptile.

r4: (is a, viviparous animal) ∧ (is a, constant temple animal) → mammal.

r5: (is a, variable temperature animal) ∧ (is a, semi-aquatic animal) ∧ (breath with, lung) → amphibian.

Algorithm 1 Extracting rules according to the data set

Require: Training record E;
Ensure: collection of relation-label value pair $A(R_i, L_i)$, an ordered set of classes $Y_0\{y_1, y_2, , y_k\}$, initial list of rules R={}
For every class y ∈ $Y_0 - y_k$ **do**
while the termination condition is not satisfied **do**
1. generate a rule r ← (A, y);
2. remove the training records covered by r from E;
3. append r to the end of the rule list R: R ← R ∪ r
End while
End For

According to the above rules, we give partial knowledge graph about the classification of vertebrates as Fig. 3 shows. Entity swallow and its labels can be represented as Ent(swallow) = Lbl(flying animal), Lbl(feather). In Fig. 3, Ent(swallow) and Ent (bird) have two matching characteristics, so we can recognize the swallow as a bird at the probability of two-thirds. The probability, denoted as P can be computed according to (1):

$$P = \frac{|Ent(E1) \cap Ent(E2)|}{|Ent(E1)|} \quad (1)$$

where $|Ent(E1) \cap Ent(E2)|$ indicates the number of common labels of entity E1 and E2, $|Ent(E1)|$ indicates the number of labels of entity E1.

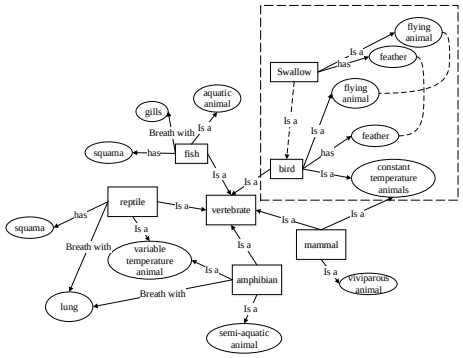


Figure 3. Partial knowledge graph of vetebrates.

4.2. Establishing New Association through Knowledge Reasoning

In knowledge graph we can establish a new association between more entities through knowledge reasoning, so as to expand relations between entities and increase the edge density of the knowledge graph. Reasoning often requires the support of relevant rules, such as "spouse" and "male" to reason out "husband", from the date of birth and the current time to reason out of age and so on. These rules can be constructed manually by people, but often time-consuming and laborious, and it is difficult to exhaust all the reasoning rules in complex relationships. At present, it mainly relies on the co-occurrence of relations, and uses association mining techniques to automatically find reasoning rules. The classical method of using relational rule to achieve relational extraction is Path Ranking Algorithm, which uses each different relational path as a one-dimensional feature. By constructing a large number of relational paths in knowledge graph, we are able to obtain the eigenvector of relation classification and a relational classifier to extract the relationship. For example, there is doubt that whether it is correct to establish a relationship profession between entity Charlotte Bronte and entity writer in Fig. 4. We can calculate the rate of correctness of a relationship, denoted as $Cr(E1, R, E2)$, according to (2):

$$Cr(E1, R, E2) = \sum_{\pi \in Q} P(E1 \rightarrow E2, \pi)\theta(\pi) \quad (2)$$

where P(E1, E2) indicates one path between E1 and E2, Q indicates all paths starting from E1 and ending with E2, $\theta(\pi)$ represents the weight obtained by training.

4.3. Semantic links and active push of information

Knowledge graph as an important support for semantic query includes a large number of named entities and semantic relations. Knowledge graph can provide an open knowledge access interface and to a certain extent it reflects the real world of inter-entity relations. The graph structure of knowledge graph is not restricted by form. Knowledge graph

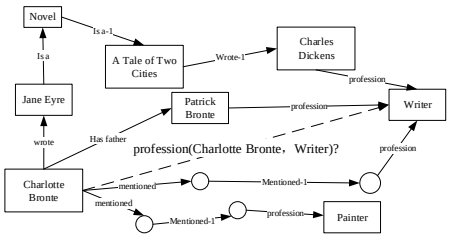


Figure 4. Partial knowledge graph of people.

can express abundant natural semantics and can supplement related information among terms. The graph-based nature of knowledge graph makes possible a linkage to other graphs thus resulting in an easy integrating of multiple kinds of information and an enhancement in integrity of information. By exploring the graph, new connections and commonalities between items and users can be discovered and exploited. For example, in Fig. 5, when a user enters the question "What happened to Air France on September 15?", we can find the corresponding entity nodes that are "Air France" and "9.15" in the knowledge graph. Then we learn that there is a staff strike event occurred in Air France on September 15. Through entity-relation links and relevant information pushing, we can get an extended knowledge graph and show the user more detail information such as flight numbers and follow-up effects of the incident.

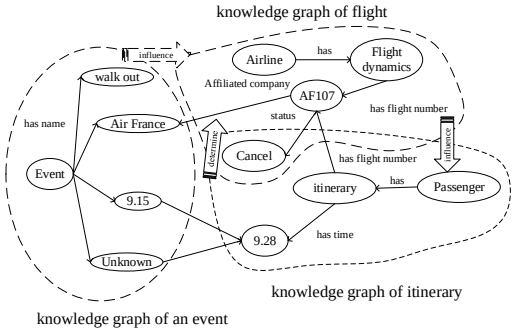


Figure 5. A combined knowledge graph.

5. Using Knowledge Graph to Answer the Question of "How"

Knowledge graph can also show a process of dealing with an event when a user enters a question "how to do...". Path queries on a knowledge graph can be used to answer compositional questions such as What languages are spoken by people living in Lisbon?. However, knowledge graphs often have missing facts (edges) which disrupt path queries. Recent models for knowledge base completion impute missing facts by embedding knowledge graphs in vector spaces. We show that these models can be recursively applied to answer path queries, but that they suffer from cascading errors. This motivates a new "compositional" training objective, which dramatically improves all models ability to answer path queries, in some cases more than doubling accuracy. Let E be a set of entities and R be a set of binary relations. A knowledge graph G is defined as a set of triples of the form (s,r,t) where s,t $\in$ E and r $\in$R. A path query q consists of an initial anchor entity, s, followed by a sequence of relations to be traversed, p = (r1,...,rk). The answer or denotation of the query, [q], is the set of all entities that can be reached from s by traversing p. Formally this can be defined recursively:

$$[s] \stackrel{\text{def}}{=} \{s\}, [q/r] \stackrel{\text{def}}{=} \{t : \exists s \in [q], (s, r, t) \in G\}$$

We define the set of candidate answers to a query C(q) as the set of all entities that "type match", namely those that participate in the final relation of q at least once. And let N(q) be the incorrect answers:

$$C(s/r1/.../rk) \stackrel{\text{def}}{=} \{t|\exists e, (e, r_k, t) \in G\}, N(q) \stackrel{\text{def}}{=} C(q)\backslash[q].$$

For example, when a user enters a "how to recruit", we can find other entities that have an "include" relationship with the entity "recruitment process" based on the keyword recruit in Fig. 6. We can then query other entities related to the previous entity follow the path if necessary and continue to perform these two steps until all relevant entities are found.

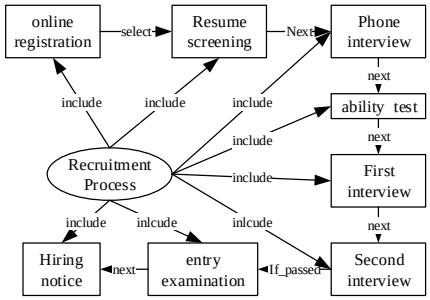


Figure 6. Partial knowledge graph of recruitment process.

6. Using Knowledge Graph to Answer the Question of "Why"

When people want to know the cause of something, she/he usually raise questions related to why, and the reason for one thing includes both direct and indirect causes. Five Whys is an iterative interrogative technique used to explore the cause-and-effect relationships underlying a particular problem [15]. The primary goal of the technique is to determine the root cause of a defect or problem by repeating the question "Why?" Each answer forms the basis of the next question. We give a description in Algorithm 2. The "Why" in the name derives from an anecdotal observation on the number of iterations needed to resolve the problem. We give an example in Fig. 7 by asking why the vehicle cannot starting. Through the first inquiry we know the reason is the battery is dead. Then we continue to ask questions

"Why the battery is dead?" Through five iterations, we can know that the root cause is that the vehicle was not maintained according to the recommended service schedule. The questioning for this example could be taken further to a sixth, seventh, or higher level, but five iterations of asking why is generally sufficient to get to a root cause. The key is to encourage the trouble-shooter to avoid assumptions and logic traps and instead trace the chain of causality in direct increments from the effect through any layers of abstraction to a root cause that still has some connection to the original problem. Note that, in this example, the fifth why suggests a broken process or an alterable behaviour, which is indicative of reaching the root-cause level. It is interesting to note that the last answer points to a process. This is one of the most important aspects in the Five Whys approach - the real root cause should point toward a process that is not working well or does not exist.

Algorithm 2 Iterative interrogative algorithm

Input: Question Q {Why Entity E_0 ...?};
Output: A directed acyclic graph G
while E_i is the root cause of E_0 **do**
1. Find the entity E_i that matches the triple $(E_0, causedBy, E_i)$;
2. i++;
3. Add node E_i and relationship $causedBy$ to G.
End while

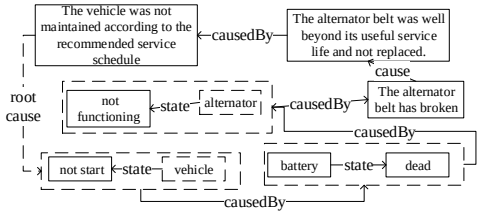


Figure 7. An example knowledge graph about causes of not starting vehicle.

In addition to inquiry the causes of the occurrence of an event, "why" can also be used to question the causality. A cause and its effect can be of different kinds of entity. For example, when a user inquiries about why smoke damage people's lungs, smoke is the cause and damage to lungs is the relevant effect. We show the detail reason through knowledge graph by finding the middle nodes. In Fig.8 we can find that there are five paths between entity smoke and entity lung. Each path indicates a reason for that smoking will damage people's lungs. In algorithm 3 we elaborate how to find all the complete paths between two entities in order to find the detailed cause of the causal relationship between them.

Algorithm 3 Finding all the complete paths between two entities

Input: Entity E1 and Entity E2;
Output: Knowledge graph G with all paths between E1 and E2
while E_2 is visited **do**
1. Find the first neighbour E_i of E1 and mark it as visited;
2. E1=E_i;
End while
3. Back to the last node;
4. Repeat step 1 and step 2.

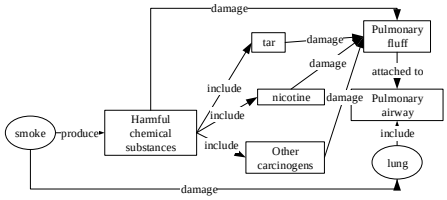


Figure 8. Causal relationship between smoke and lungs.

We expand Fig. 6 to illustrate details of the recruitment in a more detailed diagram. In Fig. 9, we can easily find the time and place of the recruitment as well as the interviewers who are responsible for the interview based on some discrete data points. These discrete data points can be the answers to "when", "where" and "who" respectively. Linking discrete data points through relevant connections can reflect some of the basic situation of the recruitment. Contents in the right green box show the recruitment process which is the answers to "how" questions. And according to contents in the purple box we can understand the reasons for holding the recruitment. Certainly, there are some other reasons not shown in the figure which yet can be inferred from the existing known knowledge. For example, reason for selecting Alice and Bob as the interviewers may be that Alice and Bob are very experienced.

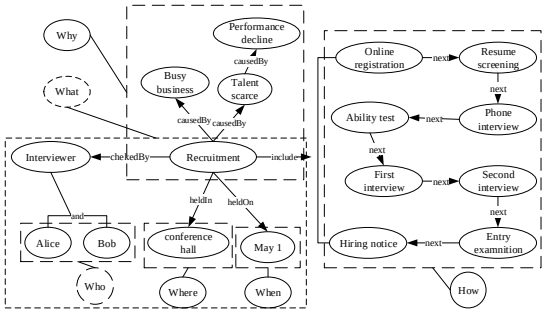


Figure 9. An example of reflecting Five Ws on a knowledge graph.

7. Related works

For users with complex information needs, they can express their needs by proposing natural language questions. These questions are subsequently interpreted with respect to the specific knowledge graph at hand by mapping them to a triple pattern query, which can be issued to the knowledge graph, returning the desired answers to the user [3, 4, 5, 8]. This new setting of large knowledge graphs presents an opportunity to tackle the question answering problem using new approaches. By working against a knowledge graph, crisp entities can be returned as answers. By exploiting the structure provided by the knowledge graph and extracting relationship between entities, one can also answer complex questions that require multiple joins, corresponding to paths

in the knowledge graph [6, 13, 14]. In these cases, a knowledge graph can be used to return answers that are proper tuples of entities, rather than singletons. Moreover, users will be able to ask for how an answer was derived by looking at the query that produced it. In [7] the authors presented a semantic parsing framework for question answering using a knowledge base and defined a query graph as the meaning representation that can be directly mapped to a logical form. Semantic parsing is reduced to query graph generation, formulated as a staged search problem. In [9] the authors presented a natural-language question-answering system that gives access to the accumulated knowledge of one of the largest community projects via an automatically acquired structured knowledge base. Inheriting concepts from ECML, at the highest abstraction level, in [11] the authors elaborated a contract in the eSourcing ontology and eSML answering three conceptual questions including the "who, where, and what" question.

8. Conclusion

The availability of large knowledge graphs presents a challenge and an opportunity that are, in some sense, duals of each other. The challenge lies in how to make an abundance of knowledge easily accessible to humans. Only a fraction of potential consumers of this knowledge will be versed in formulating triple pattern queries that express their information needs. Even for such people, the sheer size of data and the lack of a schema mean that formulating a triple pattern query can be a challenging task, requiring multiple rounds of tedious query reformulation. A solution to the above problem is to allow users to query knowledge graphs by proposing natural language questions. We proposed in this work to introduce knowledge graph to solve questions guided by the interrogative words including who/when, what, how and why(Five Ws). We elaborate three algorithms in this dissertation to facilitate effective querying of knowledge graphs and acquisition of knowledge. The contributions of this work are in the areas of question answering over knowledge graphs, relaxed knowledge graph querying combining structured and unstructured data, and open information extraction. These works initially validate that we can use natural language to express knowledge graph and then evaluate a large number of knowledge theories and systems. On the other hand, data, information, knowledge, and wisdom forms progressive layers of expression. We can dig information from data, obtain knowledge from information and acquire wisdom from knowledge. Our work lays the foundation of investigation from data to wisdom. In the next stage, we will deal with the 5W problems for the same background and different background at data, information, knowledge and wisdom level respectively. In addition, we will deal with the ambiguity of mapping problem to the query which is both structural and informative.

Acknowledgments

This paper is supported by NSFC under Grant (No.61363007, No.61502294, No.61662021, No. 61661019), NSF of Hainan No.ZDYF2017128 and No.20156243, NSF of Shanghai No.15ZR1415200, and STCSM project No.14YF1404300.** refers to corresponding author.

References

[1] M. Yahya, *Question answering and query processing for extended knowledge graphs*, 3rd ed. Clerk Maxwell, A Treatise on Electricity and Magnetism, vol. 2. Oxford: Clarendon, 1892, pp.6873.

[2] M. Joshi, U. Sawant and S. Chakrabarti, *Knowledge Graph and Corpus Driven Segmentation and Answer Inference for Telegraphic Entity-seeking Queries*. Conference on Empirical Methods in Natural Language Processing. 2014:1104-1114.

[3] K. Guu and J. Miller and P. Liang, *Traversing Knowledge Graphs in Vector Space*. Computer Science, 2015.

[4] Y. Lin and Z. Liu and H. Luan, et al., *Modeling Relation Paths for Representation Learning of Knowledge Bases*. Computer Science, 2015.

[5] J. Carroll and I. Dickinson and C. Dollin, et al., *Jena: implementing the semantic web recommendations*. International World Wide Web Conference on Alternate Track Papers & Posters. ACM, 2004:74-83.

[6] P. Minervini and N. Fanizzi and C. D'Amato, et al., *Scalable Learning of Entity and Predicate Embeddings for Knowledge Graph Completion*. IEEE, International Conference on Machine Learning and Applications. IEEE, 2015:162-167.

[7] W. T. Yih and M. W. Chang and X. He, et al., *Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base*. Meeting of the Association for Computational Linguistics and the, International Joint Conference on Natural Language Processing. 2015:1321-1331.

[8] A. Bordes and S. Chopra and J. Weston, *Question Answering with Subgraph Embeddings*. Computer Science, 2014.

[9] P. Adolphs and M. Theobald and U. Schfer, et al., *YAGO-QA: Answering Questions by Structured Knowledge Queries*. IEEE Fifth International Conference on Semantic Computing. IEEE Computer Society, 2011:158-161.

[10] Chaim Zins, *Conceptual approaches for defining data, information, and knowledge*. Journal of the American society for information science and technology, 2007, 58(4):479-493.

[11] A. Norta and L. Ma and Y. Duan, et al., *eContractual choreography-language properties towards cross-organizational business collaboration*. Journal of Internet Services and Applications, 2015, 6(1): 8.

[12] T. Aven, *A conceptual framework for linking risk and the elements of the datainformationknowledgewisdom (DIKW) hierarchy*. Reliability Engineering & System Safety, 2013, 111:3036.

[13] N. Lao and W. W. Cohen, *Relational retrieval using a combination of path-constrained random walks*. Machine Learning, 2010, 81(1):53-67.

[14] K. Xu, S. Reddy, Y. Feng, et al., *Question Answering on Freebase via Relation Extraction and Textual Evidence*. 2016.

[15] O. Serrat, *The five whys technique*. Asian Development Bank, 2009.

[16] G. Bellinger and D. Castro, *Data, information, knowledge, and wisdom*. European Congress of Rheumatology Eular. BMJ Publishing Group, 2004:276-276.

BPaaS: A Platform for Artifact-centric Business Process Customization in Cloud Computing

Min Gao,Yuyu Yin,Xingfei Wang
School of Computer, Hangzhou Dianzi University
Key Laboratory of Complex Systems Modeling and Simulation of Ministry of Education
College of Electrical Engineering, Zhejiang University
Hangzhou, Zhejiang, P. R. China
Email:{*152050068, yinyuyu, 151050148*}*@hdu.edu.cn*

Ying Li
College of Computer Science
Zhejiang University
Hangzhou, Zhejiang, P. R. China
Email:*cnliying@zju.edu.cn*

Lipeng Guo
College of Computer Science
Fudan University
Shanghai, P. R. China
Email:*lpguo@fudan.edu.cn*

Zaidie Chen
Hangzhou Housing Management Bureaus
Hangzhou, Zhejiang, P. R. China
Email:*czd@tmsf.com*

Lipeng Guo
College of Computer Science
Fudan University
Shanghai, P. R. China
Email:*lpguo@fudan.edu.cn*

Ying Li
College of Computer Science
Zhejiang University
Hangzhou, Zhejiang, P. R. China
Email:*cnliying@zju.edu.cn*

***Abstract*—As a new service paradigm of Software-as-a-Service (SaaS),Business Process-as-a-Service (BPaaS). BPaaS is used to build a cost-effective Business Process Management (BPM) system. Based on universal Artifacts, we develop a framework named SeGA (Self-Guided Artifact). In SeGA, a BPM system is capable of executing business processes from multiple clients, and responding query at runtime. what's more, by the template-based cascading data mapping method, entity to be synchronized with database automatically, so each BP instance can modify its process entity without worrying about the database access. In this paper, we conduct a deep research and implement a prototype system for Artifact process design.**

Keywords: SaaS, BPaaS, Universal Artifact, SeGA

I. MOTIVATION

A business process (BP for short) is a sequence of tasks that are performed by humans or computers to accomplish a business objective. In traditional settings, a BPM system always suffers from low adaptability, which means the BPM system is highly likely to be redesigned along with the change of environment or requirement [1].

In traditional business process modeling, activity is the "first-citizien" and the main focus is the control flow, such as resource planning and logistics, mainly aiming at business management in general instead of software design or development. As a consequence, once any change occurs, the system has to pay the price.

Many miserable stories happen, and we will give an example of the BPM system that is used in the Hangzhou Housing Management Bureaus (HHMB for short). Note that HHMB has already gained rich experience in using BPM systems. Now, for the same bureau of Urumqi (the capital city of Xinjiang province), the administrative processes are similar. However, it differs from the case of Hangzhou in organization architecture and process complexity. Thus, it is a challenge to take advantage of the existing BPM systems. Although their tenants differ in terms of specific requirements, they do share the common requirement of administrative processes.

From the example upon, it is obvious to see the missing data is a key reason for hindering software design and management. The real world is not kind, for the absence of data, any change happened may disturb the executable activity sequence. This model is not scalable. Faced up with the change whether from data or organization structure, it is technically difficult to implement the change on the prototype system in practice due to the lack of coherent data design [3]. Honestly, there are more than one example like this. Processes in real world tend to be semi/loosely structured or unstructured, knowledge-intensive and usually driven by user decision and

DOI reference number: 10.18293/SEKE2017-196

data. These processes cannot be represented as a set of activities with predefined precedence relations. Availability and status of data objects are the main driver for the process progression. So to model and implement data-driven processes, a tight integration of processes and data is required. To overcome this limitation, we introduce the Artifact-based business process[4]. The artifact-centric approach[5] emerged as a foundational proposal for merging data and processes together. In Artifact-centric business process, the change rules could be temporary and non-schematic. Rules allow business processes to respond to situations flexibly with many more options.

Recently Software-as-a-Service (SaaS) has begun to be employed in business process manage. Cloud computing has undoubtedly fired the desire to provide BP execution as service or BPaaS[2]. By providing a set of pre-designed workflows on the SaaS application，a tenant can simply select and configure the most suitable one and use the business processes execution on-demand as a Service and thus brings a new service paradigm- Business Process-as-a-Service (BPaaS). BPaaS proposes a novel way to integrate the techniques of SaaS and BPM and enables multiple tenants (customers) to share common business process model and resources among each other. This has the disadvantage that multi-tenant models are often limited in terms of customizability: one model should fit the needs of all customers[6][7][8].

BPaaS can be regarded as a special SaaS that is used in business process management. SaaS is one of different ways to provide cloud services, through providing configurable interfaces in the Internet or Intranet. Similarly, BPaaS provides BP services for multiple users in a way of service. That is, it is not necessary for users to manage or control the underlying resources. Since the BPaaS provides resources through interfaces in the Internet or Intranet, the users can conveniently access the resources in the system via a browser, and do not need to invoke any low-level fine-grained component [9]. Note that, it is also an option to use the multi-tenancy technique to build a BPM system.

Consider real estate property management in China again. There are roughly10 to 50 Housing Management Bureaus (HMBs) in each of 30 provinces for managing titles, permits, licenses etc. Every HMB currently runs and maintains its own BPM system[10]. For example, the BPM system for the HMB in a large city Hangzhou handles about 300,000 cases annually (with about 500 BP models)(Fig.1 and Fig.2). BPaaS can get an estimate 9% labor saving for Hangzhou HMB in managing and maintaining BPM systems[11] and is a great business opportunity in the software market in general.

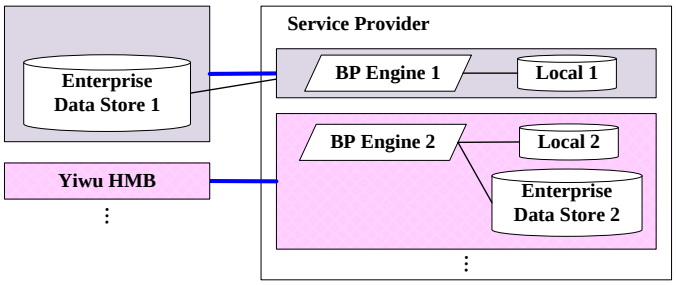


Fig. 1. Running Clients' BP Engines

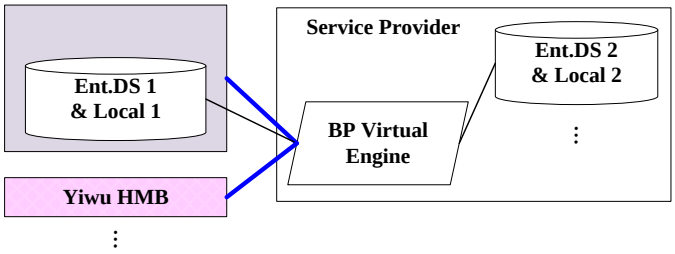


Fig. 2. Shared BP Virtual Engine

However, in most BPM systems (workflow systems), the data for process execution is scattered across databases for enterprise, auxiliary local data stores within the BPM systems, and even file systems (e.g., specification of process models). The interleaving nature of data management and BP execution and the lack of a coherent conceptual data model for all data needed for execution make it hard for (1) providing BPaaS (2) effectively support collaboration between business processes, due to an enormous effort required on maintaining both the engines as well as the data for the client applications. In particular, different modeling languages and different BPM systems make process interoperation one of the toughest challenges. In [12], Professor Su formulated a concept of a "universal artifact", which extends artifact-centric models by capturing all needed data for a process instance throughout its execution. A universal artifact clearly captures all necessary data for execution and allows for a BP engine to process without any data outside of the universal artifact. In this paper, we conduct a deep research into the BPaaS model[13] and implement a prototype system for artifact process design.

II. ARTIFACT

Artifact is a key, business-relevant conceptual dynamic entity that is used in guiding the operations of a business, and whose content evolves as it moves through those operations. It aims to provide a unified, end-to-end view of relevant entities and their possible evolutions.

A. Maintaining the Integrity of the Specifications

An artifact type consists of two parts named information model and lifecycle model, aiming to provide a unified, end-to-end view of relevant entities and their possible evolutions.

- The information model is defined as a set of attributes, and includes all the data needed to capture business process goals and evaluate whether they are achieved, for the runtime state of a business process is determined by the current snapshot of all artifacts.
- Lifecycle model is the description of the allowed information model evolutions through the execution of a process. it specifies possible ways that an artifact might progress through the business but does not restrict the way to specify artifact lifecycles.

Identify applicable sponsor/s here. *(sponsors)*

III. ARCHITECTURE

The architecture of the target BPM system is shown in Fig. 1. The Designing layer is responsible for modeling, including Process Customizer, Entity Customizer and Data Connector. We will elaborate the three components below(Fig. 3).

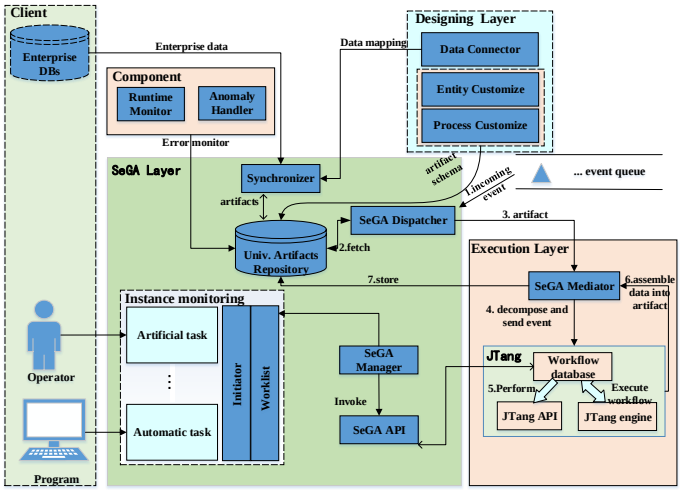


Fig. 3. The Architecture of the Target BPM System

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

A. *The Entity Customizer*

The Entity Customizer provides a series of tools to design schema and further saved in the repository in the SeGA layer. The Entity Customizer is responsible for data design. This module first imports the business process entity template of the corresponding process template, and the customer can modify the entity template and conduct attribute configurations during the designing process according to the specific business requirements. The process entity needs to consider three types of data: Enactment status which represents the current execution snapshot, resource usage and state needed for service execution and correlation between processes instances. The Artifact entity model in the BPaaS platform is a tree-shaped model[14], Each node represents an Artifact data. The following entity types are defined in this part:

Artifact: All the data nodes in this platform are of Artifact type. And entity of this type specifically refers to the one who has sub-nodes and has a one-to-one mapping with its instance. For example in Fig. 5, the Applicant Information of Public rental application form, It has sub-nodes such as name, age, but only one “applicant information” is allowed.

Artifact: 1-n: Artifact with a relationship of one-to-many. For example in Fig. 5, different from the “appliciant information”, After instantiation, the “Family member information” can matching with several members for the Multi-member family .Entity of this one-to-many relationship is set to be Artifact 1-n type.

Attribute: Nodes of the Entity tree which represents inseparable metadata, such as “name”. Specially, the Attribute attribute used to represent the uniqueness of the Artifact instance is called “Key”.

Group: Complex attribute type which stores as a metadata in the database, but in reality, it may be divided into multiple ones. For example in Fig. 4,in practical case, the “address” may be divided to “province” and “city” for use, but the separately stored attribute of province and city has no meaning in store, for this mean, the type “Group” is essential.

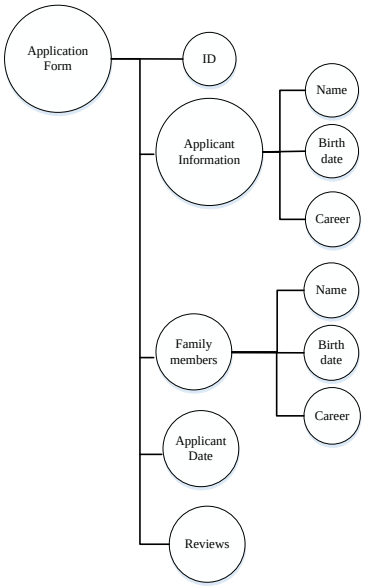


Fig. 4 Artifact Instance-Application Form

B. *The Process Customizer*

The Process Customizer adjusts the artifact lifecycle that is stored in the universal artifacts repository (UAR for short, in the SeGA layer) (Fig.6), and also makes service binding. Customers can customize the process by simply dragging the icon (such as the start node, task node, etc.) in the page according to his own need. And in service binding, by setting the read and write permissions of different fields in the business entity, the binding of the process task and the business entity is realized. The Process Customizer is in charge of setting basic activity attributes(introduction, merge mode, branch mode), as well as the attributes(conditional expressions) among different tasks. Each activity needs to specify which attributes to read and write.

C. *The Data Connector*

The Data Connector is a tool to define the mapping relationships in the entity-enterprise database (EDB for short), which are maintained by the Synchronizer in SeGA layer [14]. It includes two parts called DB-Template and EDB mapping and Entity-EDB mapping. Through the pattern mapping technology to establish relationship between the association automatically and add the mapping rules manually. Due to the different focus of business people and programmers, there are differences between the understanding of business entities and data of enterprise database, together with the heterogeneity of the data structure, it is difficult to establish a direct mapping

between the business entity and enterprise data, and thus can not guarantee the data consistency.

Below is the basic theory of cascade mapping(Fig.5). O represents cascading operations and M represents data mapping. The source mode includes business entity template S’ and business entity S, target source mode includes data-base template T’ and enterprise database T. In order to solve the problem of large differences in structure and attribute names of S and T, using the business entity template S 'and the database template T' to achieve automatic data mapping of S and S’,T and T’. between the two. In the S 'and T' mapping, the use of pattern mapping technology can automatically complete the mapping of partical data, and meanwhile, manually modified mapping rules is allowed. In this way, we can establish the data mapping relationship from S to T by establishing the mapping relation between S and S ', S' and T ', T' and T respectively.

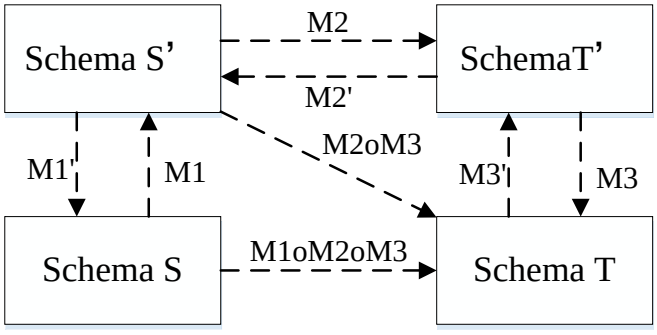


Fig. 5 cascade mapping

Based on the template-based cascading data mapping method[14], the data connector solves the differences in the understanding caused by the different points of interest by bridging the business template and the database template, and the part of the mapping work is automatically completed improving the efficiency of the mapping greatly. That is, Data mappings allow entity to be synchronized with database automatically. So each BP instance can modify its process entity without worrying about the database access[15][16].

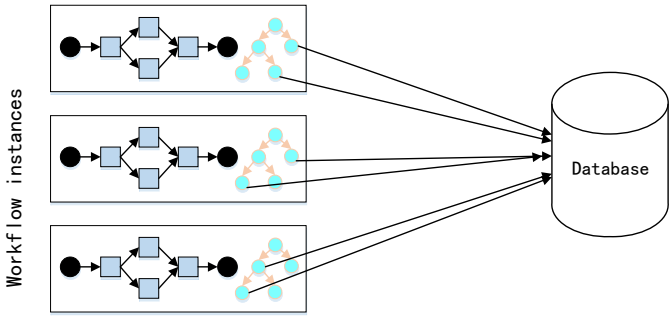


Fig. 6 data mapping

The SeGA layer acts as a Service Wrapper/Mediator [17]and seperates data from the execution engine, which is a key advantage for BPaaS [1]. It consists of a SeGA dispatcher and a SeGA mediator. When an external event arrives, the dispatcher fetches the relevant universal artifact from a universal artifact repository, extracts the schema from the universal artifact and maps it back to the original form(GSM[18][19] or EZ-Flow) restores the original artifact instance (GSM or EZ-Flow), and sends the external event, schema, and the original artifact instance to the mediator. When the mediator receives the event, schema, and the instance, it deposits the artifact schema in the appropriate location where the Barcelona/EZ-Flow engine will access, and passes the control over to the Barcelona[20]/EZ-Flow engine by forwarding the event. When the Barcelona/EZ-Flow engine receives the incoming event, it executes the next step and updates the artifact instances according to the schema deposited by the mediator; and outgoing events may also be sent directly from the engine if there exists task invocation during the execution. Once it completes, the mediator fetches the updated artifact instances, together with their schemas and states, and sends them back to the dispatcher. The dispatcher then maps the instances and schemas back to universal artifacts and stores into the corresponding repository.

The Execution layer consists of two, i.e., the SeGA mediator and JTang workflow system. The Execution layer is responsible for pure workflow management, not involving any data. The Task Manager invokes SeGA API and JTang API to connect the remote database. After loading data into the workflow system, SeGA invokes the JTang engine to execute the workflow. The JTang engine handles the event, updates the instance and sends the processed events. The JTang engine determines whether the next activity can be implemented according to the execution flow. The execution result will be received by the Task Manager component in the SeGA layer to decide which task is to perform next. There are two components that can control the task performance. 1) One is the SeGA Manager component that informs the program to execute the task. 2) The other is that people can start an initiator component to initialize the artifact instance. Besides, the Runtime Monitor component and Anomaly Handler component can handle the possible errors in the execution process in a timely manner.

IV. IMPLEMENTATION

When people use BPaaS in a real application, the proposed system provides templates for the process designer to specify their requirements. Based on the template-EDB and entity-EDB mapping, people can acquire the mapping between the schema and EDB table attributes. The proposed system stores such mapping relations into the dataset MySQL. We use the programming library JointJS1 to customize the BP template, which is displayed on the Web. We set the access permission to data for each activity according to two rules, i.e., read-only and write-only. Thus, people can acquire a complete business process, and store the web schema of each activity in the local database. We open the source code of BPaaS at https://github.com/rendididi/SeGA.

V. DEMONSTRATION

In demonstration, we set the process management of Urumqi Housing Management Bureau (UHMB for short) as a template. Due to the large population migration, we need a pre-step to make an investigation to confirm whether the residential address of the applicant changed or not. Further, different organizations have different configuration, which means

different data formats and sizes. So we need both Customize Entity and Customize Process component. We employ the following four procedures to implement the customization of BPaaS.

Entity Customization. The business experts choose the appropriate template to customize the entity, and send the request or data to the process execution engine. We can get the activity ID with some calculation, and after that, we can modify the artifact attributes of the template according to the requirement of the enterprise. For example, in the UHMB case, we need an extra attribute to confirm the current address of an applicant.

Data Mapping. This procedure selects the artifact database to map, and sets the basic information, such as type and address, to build the mapping relationship of DB Template and EDB. In detail, this procedure conducts the one-to-one mapping between an artifact property with a field in the database, and handles the automatic data synchronization in business process management. Also, the data mapping procedure modifies the processes for UHMB according to the template.

Process Customization. This procedure holds the position task modules to modify the processes of UHMB and sets the basic attributes. For each task in the process sequence, this procedure configures read and write permission for the attributes that are generated in the first procedure (Entity Customization). The system will return the corresponding forms according to the activity ID when the operators are responsible for different steps in the process steps.

Process Execution. In the Task Viewer, the operator generates an instance and sends the request to the process execution engine, which will return the activity ID with some calculation. According to the ID, this procedure generates the appropriate Web form. The system updates the database after the form is filled and submitted, and sends the form along with the ID to the execution engine. Meanwhile, the engine returns the ID of the activity that is to be executed next, and repeats the above procedure until the whole business process finishes.

The demonstration introduced in this paper is accessable via the link https://youtu.be/ms2435UyeEI.

ACKNOWLEDGMENT

The research in this paper is supported by the National Key Technology R&D Program under Grant (No.2014BAK14B04), Zhejiang Provincial Natural Science Foundation (No.LY12F02003), China Postdoctoral Science Foundation under Grant (No.2013M540492). Last but not the least, I would like to express my heartfelt gratitude to Professor Jianwen Su, who have instructed and helped me for this paper.

REFERENCES

[1] Xu, W., Su, J., Yan, Z., Yang, J., Zhang, L. An artifact-centric approach to dynamic modification of workflow execution. In: Proc. 19th Int. Conf. on Cooperative Information Systems (CoopIS), pp. 256–273,2011.

[2] Y Y. Sun, J. Su, and J. Yang. Separating execution and data management: A key to business-process-as-a-service (BPaaS). In Proc. Int. Conf. on Business Process Management (BPM), 2014.

[3] F. Casati, M. Castellanos, U. Dayal, and N. Salazar. A generic solution for warehousing business process data. In Proceedings of the 33rd international conference on Very large data bases, VLDB '07, pages 1128–1137. VLDB Endowment, 2007.

[4] K.Bhattacharya,C.Gerede,R.Hull,R.Liu,andJ.Su.Towards formal analysis of artifact-centric business process models. In Proc. Int. Conf. on Business Process Management (BPM), Brisbane, Australia, September 2007.

[5] Nigam A, Caswell N S. Business artifacts: An approach to operational specification[J]. Ibm Systems Journal, 2003, 42(3):428-445.

[6] Chang, S.H., Kim, S.D.: A Variability Modeling Method for Adaptable Services in Service-Oriented Computing. In: Proc.11th International Software Product Line Conference, , pp. 261–268, SPLC 2007.

[7] Koning, M., ai˜Sun, C., Sinnema, M., Avgeriou, P.Vxbpel: Supporting variability for web services in bpel. Information and Software Technology 51,pp. 258–269 ,2009.

[8] Mietzner, R., Metzger, A., Leymann, F., Pohl, K.: Variability modeling to support customization and deployment of multi-tenant-aware software as a service applications. In: Proc. ICSE Workshop on Principles of Engineering Service Oriented Systems. Washington, USA, pp. 18–25. ,2009.

[9] R. Accorsi. Business Process as a Service: Chances for Remote Auditing. In COMPSAC Workshops, pages 398–403, 2011

[10] F. Casati, M. Castellanos, U. Dayal, and N. Salazar. A generic solution for warehousing business process data. In: Proc. 33rd international conference on Very large data bases, VLDB '07, pages 1128–1137. VLDB Endowment,2007.

[11] L. Limonad, D. Boaz, R. Hull, R. Vacul´ın, and F. T. Heath. A Generic Business Artifacts Based Authorization Framework for Cross-Enterprise Collaboration. In SRII Global Conference, pages 70-79, 2012.

[12] Sun Y, Su J, Yang J. Universal Artifacts: A New Approach to Business Process Management (BPM) Acm Transactions on Management Information Systems, 7(1):3,2016.

[13] Pathirage, M., Perera, S., Kumara, I., Weerawarana, S. A multi-tenant architecture for business process executions. In: Proc. IEEE International Conference on Web Services (ICWS), pp. 121–128,2011.

[14] Y. Sun, J. Su, B. Wu, and J. Yang. Modeling Data for Business Processes. In Proc. Int. Conf. on Data Engineering, ICDE '14, 2014.

[15] G. Liu, X. Liu, H. Qin, J. Su, Z. Yan, and L. Zhang. Automated realization of business workflow specification. In: Proc. ICSOC/ServiceWave Workshops, pages 96–108, 2009.

[16] W. Xu, J. Su, Z. Yan, J. Yang, and L. Zhang. An artifact-centric approach to dynamic modification of workflow execution. In Proc. of the 19th International Conference on Cooperative Information Systems, CoopIS '11, pages 256–273, 2011.

[17] Y. Sun, W. Xu, J. Su, and J. Yang. SeGA: A Mediator for Artifact-Centric Business Processes. In: Proc. Int. Conf. on Cooperative Info. Sys., CoopIS '12, pages 658–661, 2012.

[18] R. Eshuis, R. Hull, Y. Sun, and R. Vaculin. Splitting GSM Schemas: A Framework for Outsourcing of Declarative Artifact Systems. In BPM, pages 259-274,2013.

[19] R. Hull, E. Damaggio, and et al. Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events. In Proc. Of the 5th ACM International Conference on Distributed event-based system, DEBS 11, pages 51–62, 2011.

[20] T. Heath, D. Boaz, M. Gupta, R. Vacul´ın, Y. Sun, R. Hull, and L. Limonad.Barcelona: A Design and Runtime Environment for Declarative Artifact-Centric BPM. In Proc. of the 11th Int. Conf. on Service-Oriented Computing, ICSOC,pages 705–709, 2013.

Applying Probability Model to The Genetic Algorithm Based Cloud Rendering Task Scheduling

Guobin Zhang[1,3], Huahu Xu[1,3]
1. School of Computer Engineering and Science, Shanghai University, 200444 Shanghai, P.R. China
3. Shanghai Shang Da Hai Run Information System Co., Ltd, Shanghai 200444, mr_zhang2011@163.com

Honghao Gao[1,2] , Ankang Liu[4]
2. Computing Center, Shanghai University, 200444 Shanghai, P.R. China
4. Department of Mathematics, Shanghai University, 200444 Shanghai, P.R. China

***Abstract*—There are huge amount of tasks and data to be processed in cloud rendering environment. How to effectively schedule them is the key to ensure the overall performance of the cloud rendering environment. In this paper, an improved task scheduling algorithm based on genetic algorithm (PMGA) and probabilistic model is proposed, which aims to minimize the total time and cost of task scheduling . First, the fitness function relating to the total time and task cost is improved under the consideration of the user's satisfaction to rendering services. Then, a probability model is constructed for the scheduling algorithm, which is used to achieve the non-linear adaptive adjustment of the crossover rate function and the mutation rate function. As a result, the evolutionary ability of poor individuals in the population can be enhanced, avoiding the stagnation in the early stages. Finally, experiments are performed to demonstrate that PMGA has a better ability of optimization than that of traditional adaptive genetic algorithm (AGA). Our approach contributes to reduce the total time and the scheduling cost of the cloud rendering tasks.**

Keywords-Cloud Rendering; Task scheduling; Genetic Algorithm; The Fitness Function; Probability Model; The Crossover Rate; The Mutation Rate

I. INTRODUCTION

Due to the advantages of cloud computing, cloud rendering is considered as a promising application, focusing on the rendering services and providing the computing and data resources. The basic principle of cloud rendering is that the cloud rendering system divides tasks submitted by users into some independent subtasks, and dynamically allocates the subtasks to resource nodes based on appropriate scheduling strategies. After all tasks are completed, the rendering results will be merged together and returned to the users [1,2].

Most cloud computing platforms adopt MapReduce [3] to support distributed computing. Many studies have shown that the problem of task scheduling under cloud environment is an NP-complete problem [4]. Cao et.al., [5] proposed a task scheduling optimization algorithm based on ABC algorithm, which reduces the total time of completion task. Xu et.al., [6] proposed a multi-adaptive particle swarm optimization strategy for task scheduling in cloud, which has taken into account the task completion time, the operating cost and the balancing of resource load. Jie et.al., [7] proposed a genetic simulated annealing algorithm for task scheduling with dual fitness, which effectively balances the demands of the users and improve the users' satisfaction. Liu et.al., [8] proposed a task scheduling algorithm based on genetic algorithm and ant colony algorithm, which improves the speed of getting the optimal solution.

However, due to the heterogeneity of the cloud computing platform, any single intelligent group algorithm is easy to fall into the local optimum, premature and other defects. The improvement is mainly focused on the nature of the algorithm itself, ignoring the guiding role of the information in the process of evolution. The adaptability of these algorithm are not robust enough, and the scheduling time and cost can not be balanced well. The efficient task scheduling algorithm is still a long-term challenge in the research of cloud computing.

Genetic algorithm was proposed by Professor John Holland of the University of Michigan in the 1970s. It is commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover and selection [9]. The basic principle is to apply a coding technique to the individual in the population, then simulate the evolutionary process of the population. The standard genetic algorithm (SGA) usually has only one population, the crossover rate and the mutation rate are fixed ones, resulting in its slow convergence.

Srinivas et.al., [10] proposed an adaptive genetic algorithm (AGA). It took a dynamic automatic adjustment for crossover rate and mutation rate, which is based on the fitness value and the convergence level of the current group. This algorithm improved the convergence speed of the genetic algorithm, however, it is slightly slow at the beginning of the evolution process. Beyond that, only one fitness function is used to measure the adaptation of individual to different environment.

In order to improve the efficiency of task scheduling in the cloud rendering platform, an improved adaptive genetic algorithm based on probability model is proposed. The fitness function is established for the total time and total cost of the rendering tasks, which aims to achieve the minimum time and minimum cost. Then, the crossover rate function and mutation rate function is improved by applying the probability model, which effectively retains the excellent individuals in the evolution process of population and enhances the crossover rate and mutation rate of poor individuals.

The rest of this paper is as follows: In Section II, it describes the problem of tasks scheduling in cloud rendering environment, and gives the allocation pattern of tasks and the calculation method of total completion time and total cost of cloud tasks. In

DOI reference number: 10.18293/SEKE2017-193

Section III, the implementation and process of PMGA are introduced in details. In Section IV, PMGA and AGA are compared and analyzed under the same experimental environment. The experimental results proves that the performance of cloud rendering is improved by using PMGA for task scheduling. Finally, in Section V, we conclude this research and discuss future works.

II. DESCRIPTIONS ABOUT THE CLOUD RENDERING TASK SCHEDULING

Cloud rendering task scheduling is to distribute the tasks to the compute nodes by applying appropriate scheduling strategies, which enables to minimize the time, lower the cost, and keep the load balancing of resources [11].

Assuming that the cloud rendering system user submits T tasks, the system has R resources. Then, the total subtask number of the task t is defined as $taskNum(t)$, the total subtask number is defined as $subTaskNum$. Then

$$subTaskNum = \sum_{t=1}^{T} taskNum(t) \tag{1}$$

Subtasks are numbered according to the sub-sequence numbering method. If the sequence number of jth subtask of ith task is m, then

$$m = \sum_{k=1}^{i-1} taskNum(k) + j \tag{2}$$

The ETC matrix [12] is used to record the execution time of each sub task, the ECC matrix is used to record the execution cost of each sub task. Consequently, the total time and total cost of all the sub tasks are as follows:

$$F_{time(x)} = \max_{r=1}^{R} \sum_{i=1}^{n} Time(r,i) \tag{3}$$

$$F_{cost(x)} = \max_{r=1}^{R} \sum_{i=1}^{n} Cost(r,i) \tag{4}$$

Where $Time(r,i)$ and $Cost(r,i)$ represents the execution time and cost of ith subtask on rth resource, n is the subtask number on rth resource. $Cost(r,i)$ is defined as follows:

$$\begin{aligned} Cost(r,i) = {} & w_{cpu} Cost_{i\text{-}cpu}(r,i) + w_{menmory} Cost_{i\text{-}menmory}(r,i) + \\ & w_{storage} Cost_{i\text{-}storage}(r,i) + w_{bandwidth} Cost_{i\text{-}bandwidth}(r,i) \end{aligned} \tag{5}$$

Where $Cost_{i-cpu}$, $Cost_{menmory}$, $Cost_{storage}$ and $Cost_{bandwidth}$ are the cost of *CPU*, *Memory*, *Storage* and *Bandwidth*, respectively. w_{cpu}, $w_{menmory}$, $w_{storage}$ and $w_{bandwidth}$ are weighting coefficients of the four kind of resource costs, respectively. They are both in [0,1], and satisfy the following equation:

$$w_{cpu} + w_{menmory} + w_{storage} + w_{bandwidth} = 1 \tag{6}$$

The value of weighting coefficient can be set differently to measure the user satisfaction of scheduling results. They are all fixed to 0.25 in our experiments to balance the resources costs.

III. TASK SCHEDULING OF PMGA IN CLOUD RENDERING

A. Encoding and Decoding

As for the solution of cloud task scheduling based on PMGA, it needs to establish a mapping relationship between the solution of the problem and the individual of the algorithm. The current encoding modes of chromosome always include direct encoding and indirect encoding. This paper adopts indirect encoding mode to encode the node occupied by each sub task, taking into account the division of the operation under cloud rendering. The length of the chromosome depends on the number of sub tasks, therefore, a chromosome corresponds to a task scheduling strategy. The length of the individual chromosome equals to the total number of tasks, the value of the chromosome gene is the cloud resource number occupied by the task at that location.

For example, if users submit 3 tasks in a certain time, the resource number of cloud rendering system is 3. Task 1 is divided into 1, 2, 3 three subtasks. Task 2 is divided into 4,5,6 three subtasks, Task 3 is divided into 7,8,9,10 four subtasks. The chromosome length is 10, the value of each gene is [1,3]. Then The chromosome {1,2,1,2,1,3,2,1,2,3} is a feasible scheduling strategy as shown in Table I.

TABLE I. ENCODING MODE OF CHROMOSOMES

Subtask	1	2	3	4	5	6	7	8	9	10
Resource	1	2	1	2	1	3	2	1	2	3

It is necessary to decode them to obtain the distribution of the subtasks on each node. After that, the subtask performed on resource 1 is {1,3,5,8}, the subtask executed on resource 2 is {2,4,7,9}, the subtask executed on resource 3 is {6,10}.

B. Initialization of the Population

Assuming that the initial size of the population is $SCALE$, the total number of subtask is M, the number of resources is $RESOURCE$, the initialization of the population can be described as follows: the system randomly produce $SCALE$ chromosomes, the length of the chromosome is M, the values of genes are $[1, RESOURCE]$, the chromosomal gene is generated randomly within this range.

C. Double Fitness Function Model

The fitness function generates a fitness value of each chromosome. The value indicates how suitable for solving a task scheduling problem a chromosome is. The fitness function for SGA and AGA is only based on execution time of tasks. But, task scheduling problems in cloud rendering are different from general task scheduling problems because rendering services in cloud rendering environment are offered between cloud users and providers. Therefore, this paper defines $FIT_{time}(i)$ as the fitness function of the total completion time of the tasks, and $FIT_{cost}(i)$ as the fitness function of the total cost of the tasks. The calculation formula of this two function is as follows:

$$FIT_{time}(i) = \frac{1}{1 + Time_i(R_j)} \tag{7}$$

$$FIT_{cost}(i) = \frac{1}{1 + Cost_i(R_j)} \tag{8}$$

Where R_j is the jth resource of the ith individual, $Time_i R(j)$ is the completion time of the ith individual. $Cost_i R(j)$ is the cost of the ith individual. Formula (9) is used as the fitness function of the algorithm.

$$Fit(i) = w_{time} FIT_{time}(i) + w_{cost} FIT_{cost}(i) \tag{9}$$

In the formula, w_{time} and w_{cost} are weighting coefficients, and $w_{time} + w_{cost} = 1$, which can be set according to the user's needs. w_{time} and w_{cost} are both set to 0.5 in our experiments in

order to balance the proportion of the total time and the total cost of the task.

D. Probability Model and Generic Operation

1)Individual Selection Based On Double Fitness Function Model

This paper adopts the roulette wheel selection method [13] to do the selection operation. The selection probability of each individual was calculated according to the fitness function (7) and fitness function (8), and give the equations:

$$P_1(i)=\frac{FIT_{time}(i)}{\sum_{j=1}^{SCALE} FIT_{time}(j)} \tag{10}$$

$$P_2(i)=\frac{FIT_{Cost}(i)}{\sum_{j=1}^{SCALE} FIT_{Cost}(j)} \tag{11}$$

When choosing the next generation of individuals, one of the probability is selected from P_1 and P_2 to choose populations with probability c_1 and c_2 respectively, and $c_1+c_2=1$. In this way, the population both contains individual with shortest completion time and minimum cost, which provides good foundation for the next generation. c_1 and c_2 are both set to 0.5 in our experiments.

2)Improved crossover operation and mutation operation based on the probability model

Crossover operation is to match the individuals from the population randomly in pairs, and exchange their partial gene with a probability by following a particular probability rule for each individual. The mutation operation is to change the genetic value of a certain locus on each individual in the population to other alleles at a certain probability.

The crossover rate function P_c and the mutation rate function P_m determine the speed of updating and searching for the population. In SGA, P_c and P_m are both constant values. SGA is inefficient and is easy to be precocious for complex multivariable optimization problem. AGA had made some improvements based on the fitness value [10]. P_c and P_m in AGA are defined as follows:

$$P_C=\begin{cases}\dfrac{k_1(f_{\max}-f')}{(f_{\max}-f_{avg})}, & f'\geq f_{avg}\\ k_3, & f'<f_{avg}\end{cases} \tag{12}$$

$$P_m=\begin{cases}\dfrac{k_2(f_{\max}-f)}{(f_{\max}-f_{avg})}, & f\geq f_{avg}\\ k_4, & f<f_{avg}\end{cases} \tag{13}$$

Where $f_{\max}$ is the maximum fitness of the population, f_{avg} is the average fitness of the population, f' is the greater fitness of the two individuals which are to have cross operation, f is the fitness value of the two individual that are to have mutation operation. k_1, k_2, k_3, k_4 are [0,1]. In literature [10], $k_1=k_3=1.0$, $k_2=k_4=0.5$.

P_c and P_m in AGA perform a linear change according to the average fitness and maximum fitness of individual as shown in Fig. 1 and Fig. 2. P_c and P_m are equal to zero when the fitness value is relatively large. However, at the beginning of the algorithm, the individuals with higher fitness in the population may not be the optimal solution. Thus, it may lead to the occurrence of a local optimal solution.

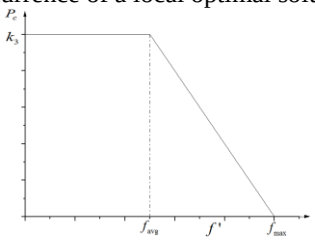


Figure 1. The Crossover Rate of AGA

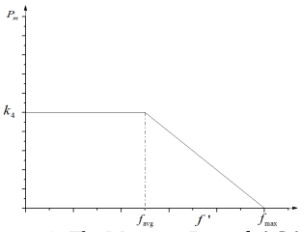


Figure 2. The Mutation Rate of AGA

The probability calculated by the probability model is applied to the crossover rate function and the mutation rate function in PMGA. The definition of the crossover rate and mutation rate function in PMGA are as follows:

$$P_C=\begin{cases}k_{c1}*\left(P_{ck}-\left|(P_{ck}-P_{pro-c})\right|\left(M_c+\frac{(f-f')}{(f_{\max}-f_{avg})}\right)\right), & \frac{f_{avg}+f_{\max}}{2}<f'\leq f_{\max}\\ k_{c2}*\left(P_{ck}-\left|(P_{ck}-P_{pro-c})\right|\left(M_c+\frac{(f_{\max}-f')}{(f_{\max}-f_{avg})}\right)\right), & f_{avg}<f'\leq\frac{f_{avg}+f_{\max}}{2}\\ P_{ck}, & f'\leq f_{avg}\end{cases} \tag{14}$$

$$P_m=\begin{cases}k_{m1}*\left(P_{mk}-\left|(P_{mk}-P_{pro-m})\right|\left(M_m+\frac{(f_{\max}-f)}{(f_{\max}-f_{avg})}\right)\right), & \frac{f_{avg}+f_{\max}}{2}<f\leq f_{\max}\\ k_{m2}*\left(P_{mk}-\left|(P_{mk}-P_{pro-m})\right|\left(M_m+\frac{(f_{\max}-f)}{(f_{\max}-f_{avg})}\right)\right), & f_{avg}<f\leq\frac{f_{avg}+f_{\max}}{2}\\ P_{mk}, & f\leq f_{avg}\end{cases} \tag{15}$$

Where $f_{\max}$, f_{avg}, f', f are the same as them in equation (12) and equation (13). k_{c1}, k_{c2}, k_{m1}, k_{m2} are constant values in [0,1], they are used to control the control the changing speed of P_c and P_m. P_{pro-c} and P_{pro-m} represents the value of the cross operation probability and the mutation operation probability calculated by the probability model. M_c and M_m are constant values in $[1,+\infty]$. The larger value they are, the higher crossover rate and the mutation rate the optimal individual has. When the fitness value is smaller than f_{avg}, the crossover rate is P_{ck}, the mutation rate is P_{mk}. The value of k_{c1}, k_{c2}, k_{m1}, k_{m2}, P_{ck} and P_{mk} can be adjusted according to the practical problems. The graph of P_c and P_m in PMGA are shown in Fig. 3 and Fig. 4, respectively.

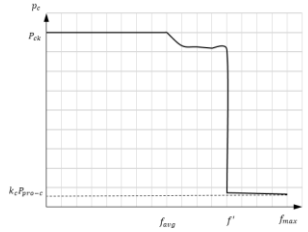


Figure 3. The Crossover rate of PMGA

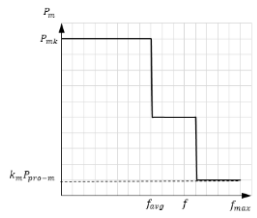


Figure 4.The Crossover rate of PMGA

When $f \le f_{avg}$, P_c and P_m in PMGA is a relatively large fixed value, enabling the individuals to evolve faster. When $f_{avg} < f \le \frac{f_{avg} + f_{\max}}{2}$, P_c and P_m decrease slowly with the increase of fitness value. When $\frac{f_{avg} + f_{\max}}{2} < f \le f_{\max}$, P_c and P_m become smaller rapidly and decrease with the increase of fitness value, when the fitness reaches its maximum value, the value of P_c and P_m are equals to $K_c P_{pro-c}$ and $K_m P_{pro-m}$, respectively, which are very close to zero. Therefore, the crossover rate and mutation rate in PGMA both have the ability to find the best solution quickly and the randomness property. When the fitness is relatively large, the value of P_c and P_m are not an absolute zero value, so that the algorithm will not be in a low searching speed in the early process of evolution.

3)Generating and Computing of Probability Model

Probabilistic model refers to a type of random system models with Markov characteristics, which is proposed by the Russian scientist Markov in 1970. According to the state space and the nature of the parameters, it can be divided into discrete time Markov chain, continuous time Markov chain, Markov decision process.

Discrete time Markov chain is a kind of Markov random process with discrete state space and discrete time. It can be defined as: $D = \{S, S_{init}, P, L\}$, S is a set of finite nonempty states, indicates all possible configuration status of the modeled system. $P \times S \to [0,1]$ is the migration probability function matrix, represents the probability of the migration between the modeled system states, $\forall s \in S, \sum_{s' \in S} P(s', s) = 1$. $L : S \to 2^{AP}$ is a label function with atomic propositions.

The process of PMGA is a continuous random process of selection, crossover and mutation operations. It is related to the state of the current population, regardless of the previous population status. In addition, PMGA is homogeneous, there exists a limit probability distribution in PMGA, and the transition probability is independent of time. As a result, discrete time Markov chain (DTMC) is chosen to model it.

The adaptive genetic algorithm based on probabilistic model is described as follows:

1) Initialize computing node of cloud rendering system rendering, go to 2)

2) Initialize parameters of cloud rendering task scheduling, then go to 3).

3) Encode parameters into chromosomes.

4) Initialize the population, go to 6).

5) The cloud rendering system error or program exception.

6) Calculate the fitness of each individual and determine whether the individual satisfies the optimal condition. Then go to step 12), if not, go to 7).

7) Do select operation, that is, according to the individual selection rules, select some excellent individuals from the current population as the new population, then go to 8).

8) Calculate the crossover rate P_c, do cross operation with probability P_c, go to 9), skip cross operation with probability 1- P_c go to 10).

9) Do crossover operation, and generate a new population, then go to 10).

10) Calculate the mutation rate P_m, do mutation operation with probability P_m, go to 11), skip mutation operation with probability 1- P_m, go to 6).

11) Do mutation operation, and generate a new population, then go to 6).

12) Decode the chromosome and get the optimal solution.

According to the above process description, the state transition diagram of PMGA is shown in Fig. 5.

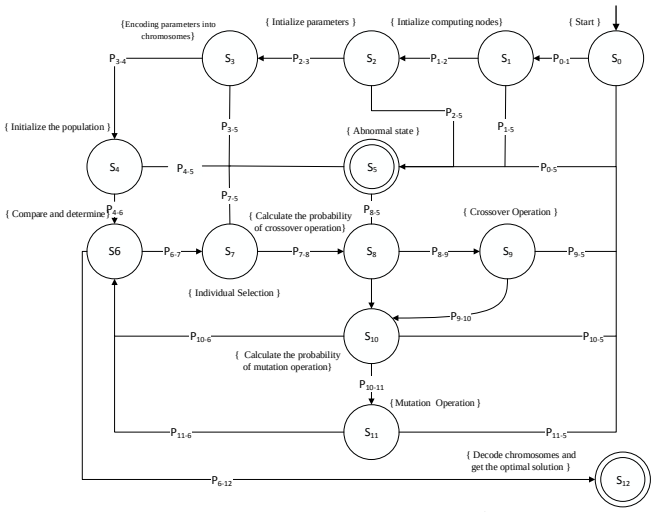


Figure 5. The State Transition Diagram of PMGA

PMGA has the properties of discrete time Markov chain. It is defined as a discrete time Markov chain D, $D = (S, S_{init}, P, L)$, each element is described as follows:

- $S = \{S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12},\}$.
- $S_{init} = S_0$
- $$P = \begin{Bmatrix} 0 & P_{0-1} & 0 & 0 & 0 & P_{0-5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & P_{1-2} & 0 & 0 & P_{1-5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & P_{2-3} & 0 & P_{2-5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & P_{3-4} & P_{3-5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & P_{4-5} & P_{4-6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & P_{6-7} & 0 & 0 & 0 & 0 & P_{6-12} \\ 0 & 0 & 0 & 0 & 0 & P_{7-5} & 0 & 0 & P_{7-8} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & P_{8-5} & 0 & 0 & 0 & P_{8-9} & P_{8-10} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & P_{9-5} & 0 & 0 & 0 & 0 & P_{9-10} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & P_{10-5} & P_{10-6} & 0 & 0 & 0 & 0 & P_{10-11} & 0 \\ 0 & 0 & 0 & 0 & 0 & P_{11-5} & P_{11-6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix} and \begin{Bmatrix} P_{0-1} + P_{0-5} = 1 \\ P_{1-2} + P_{1-5} = 1 \\ P_{2-3} + P_{2-5} = 1 \\ P_{3-4} + P_{3-5} = 1 \\ P_{4-5} + P_{4-6} = 1 \\ P_{6-7} + P_{6-12} = 1 \\ P_{7-5} + P_{7-8} = 1 \\ P_{8-6} + P_{8-5} + P_{8-10} = 1 \\ P_{9-5} + P_{9-10} = 1 \\ P_{10-5} + P_{10-6} + P_{10-11} = 1 \\ P_{11-5} + P_{11-6} = 1 \end{Bmatrix}$$
- $$L = \begin{cases} L(S_0) = \{Start\} \\ L(S_1) = \{Intialize\ computing\ nodes\} \\ L(S_2) = \{Intialize\ parameters\} \\ L(S_3) = \{Encoding\ parameters\ \text{inf}\ o\ chromosome\} \\ L(S_4) = \{Initialize\ the\ population\} \\ L(S_5) = \{Abnormal\ state\} \\ L(S_6) = \{Compare\ the\ fitness\ values\ and\ determine\ whether\ get\ the\ optimal\ solution\} \\ L(S_7) = \{Individual\ Selection\} \\ L(S_8) = \{Calculate\ the\ probability\ of\ crossover\ operation\} \\ L(S_9) = \{Crossover\ Operation\} \\ L(S_{10}) = \{Calculate\ the\ probability\ of\ mutation\ operation\} \\ L(S_{11}) = \{Mutation\ Operation\} \\ L(S_{12}) = \{Decode\ chromosomes\ and\ get\ the\ optimal\ solution\} \end{cases}$$

The probability of the boundary reachability in the Markov chain is defined as the probability of reaching the state set T

from state S within K step transition. It is expressed as: $ProbReach^{\le k}(S,T)$, The formula is expressed as follow:

$$ProbReach^{\le k}(S,T)=\begin{cases}1, k\in T\\ 0\ , k\ =0\ \&\ s\notin T\\ \sum_{s'\in S}P(s,s')*ProbReach^{\le k-1}(s',T), k\ >0\ \&\ s\notin T\end{cases} \quad (16)$$

In probability model, $P_{\text{pro}-c}$ is the probability from state S_0 to S_9, $P_{\text{pro}-m}$ is the probability from state S_0 to S_{11}. The probability between states in the model can be obtained by the running log of the cloud rendering system. The formula of $P_{\text{pro}-c}$ and $P_{\text{pro}-m}$ is:

$$P_{pro-c}=ProbReach^{\le k}\left(S_0,T\{S_9\}\right) \quad (17)$$

$$P_{pro-m}=ProbReach^{\le k}\left(S_0,T\{S_{11}\}\right) \quad (18)$$

IV. EXPERIMENTS AND ANALYSIS

A. Experimental Environment

The experiments are based on the comprehensive cloud computing service platform, which is made up of five clusters, consisting of 25 compute nodes. The tasks scheduled in the experiment are mainly the rendering of pictures and video key frames. The configuration of cloud rendering system compute node is shown in Table II.

TABLE II. COMPUTE NODE CONFIGURATION PARAMETERS OF CLOUD RENDERING SYSTEM

Cluster	The number of nodes	Operating system	CPU	GPU	Memory
1	6	Cent OS	i3-4160	GT720	4GB
2	6	Cent OS	i5-4460	GT720	8GB
3	5	Cent OS	i5-4590	GT720	8GB
4	5	Windows	i5-6500	GT720	4GB
5	3	Windows	i5-7500	GT960	8GB

B. Experimental Results and Performance Analysis

In experiment 1, the total time and total completion cost using AGA and PMGA for task scheduling are compared. Then, the average fitness value of the current new population in AGA and PMGA are compared.

The initial conditions of experiments are shown as below: (1) The number of resource is 25, the number of task is 100. (2) Each task is divided into n subtask, n is in [20,80].

The termination conditions of experiment are as follows: (1) If the number of iterations reaches the maximum G_{max} (G_{max} = 220), then the algorithm is considered to be terminated. (2) If the completion time and the total cost of continuous 40-generation are not changed, then the algorithm will be terminated.

The main parameters of PMGA and AGA in the experiment is shown in Table III.

TABLE III. MAIN PARAMETERS OF THE PMGA AND AGA

PMGA		AGA	
w_{cost}	0.50	k_1	1.00
w_{time}	0.50	k_2	0.50
k_{c1}	0.10	k_3	1.00
k_{c2}	0.50	k_4	0.50
k_{m1}	0.10	-	-
k_{m2}	0.50	-	-
$P_{\text{pro}-c}$	0.40	-	-
$P_{\text{pro}-m}$	0.08	-	-
P_{ck}	0.90	-	-
P_{mk}	0.10	-	-
M_c	1.00	-	-
M_m	1.00	-	-

Fig. 6 and Fig.7 show the iterative process of optimal total time and lowest total cost using AGA and PMGA. In Fig. 6 and Fig. 7, the horizontal axis represents the number of iterations. In Fig. 6, the vertical axis represents the task total time. In Fig. 7, the vertical axis represents the total cost. In the early evolution process, the convergence speed of AGA is relatively slow, with the further evolution, AGA only focuses on the total time, resulting in the loss of some potentially good genes. However, PMGA has a larger crossover rate and mutation rate for poor individuals, and the better individuals still have a small crossover rate and mutation rate. At the later stage of evolution, PMGA continues to evolve and eventually converges to the global optimal solution. While AGA still converges to the local optimum. The task completion time of PMGA is smaller than AGA, and the cost is significantly less than AGA. The experimental results show that PMGA reduces the total time and lowers the cost of the scheduling, which is an effective algorithm for the cloud rendering task scheduling.

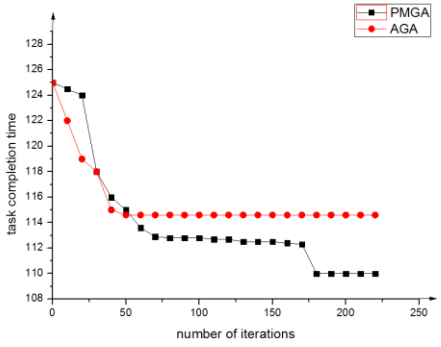


Figure 6. The Total Time for Task Completion

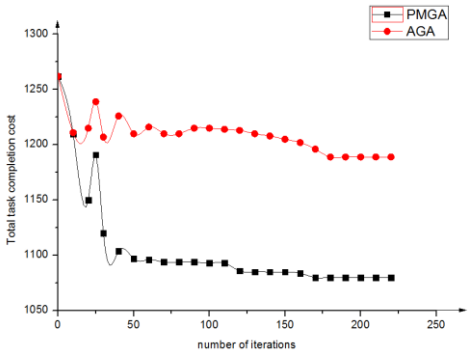


Figure 7. The Total Cost for Task Completion

Figure 8 shows the comparison of the average fitness value of the new population during the process of iteration. The horizontal axis represents the number of evolution times, and the vertical axis represents the average fitness value. AGA starts to premature convergence at about 60th generation, while PMGA continues to evolve, because when the average fitness of individuals is close to the maximum fitness, the crossover rate and mutation rate are not a zero value. Eventually, PMGA has a larger fitness value than AGA. The experimental results show that PMGA has a better ability to do the global searching.

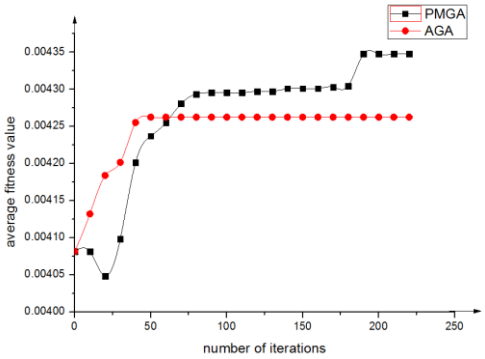


Figure 8. The Average Fitness Values of Population

In experiment 2, the number of convergence using PMGA and AGA for task scheduling are compared. The algorithm parameters are consistent with those in Table III. PMGA and AGA are compared for 50 times under the same iterations. The average value of each iterations is regarded as the ultimate result. Fig. 9 is the comparison chart of the number of convergence times using AGA and PMGA.

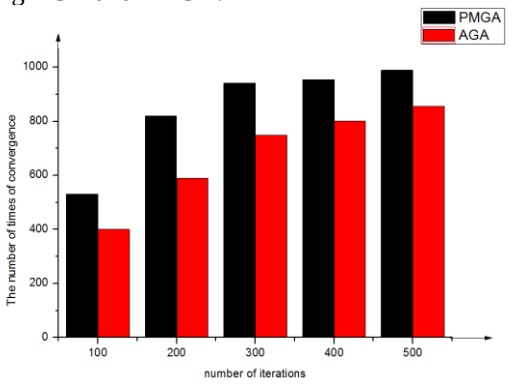


Figure 9. The Number of Convergence Using PMGA and AGA

In Fig. 9, the horizontal axis represents evolution times, the vertical axis represents the convergence times. It can be seen from the figure that the convergence number of PMGA is much more than AGA under the same number of iterations. The experimental results show that PMGA can improve the performance and robustness of the cloud rendering task scheduling.

V. CONCLUSION

This paper aims to reduce the total time and total cost of task scheduling in cloud rendering system, which applies probability model to genetic algorithm for rendering task scheduling. The algorithm evaluates the excellent degree of the new solution by establishing the fitness function considering the total time and total cost of the task. As a result, the better individuals will be returned, and the diversity of the population will be improved. The crossover rate function and mutation rate function are improved, which enhances the local search ability and avoids the occurring of local convergence. The experimental results show that PMGA reduces the total time and minimize the total cost. Thus, it is an effective algorithm for scheduling tasks in cloud rendering system.

In the future work, the Markov characteristic of PMGA will be studied in order to verify the scientificity and rationality of scheduling algorithm using PMGA. Then, we will focus on the load balancing of dynamic task scheduling under cloud rendering system environment, and consider the influence of data distribution, service quality and other impact factors on the task scheduling results.

ACKNOWLEDGMENT

This paper is supported by National Natural Science Foundation of China under Grant No. 61502294, Natural Science Foundation of Shanghai under Grant No.15ZR1415200, CERNET Innovation Project under Grant No.NGII20160325, and Foundation of Science and Technology Commission of Shanghai Municipality under Grant No.14590500500.

REFERENCES

[1] Gupta R. Above the Clouds: A View of Cloud Computing[J]. Eecs Department University of California Berkeley, 2012, 53(4):50-58.

[2] Calheiros R N, Ranjan R, Beloglazov A, et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, ‖ Software: Practice and Experience[J]. Software Practice & Experience, 2010, 41(1):23–50.

[3] Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters.[J]. In Proceedings of Operating Systems Design and Implementation (OSDI, 2004, 51(1):107-113.

[4] Ullman J D. NP-complete scheduling problems. J Comput Syst Sci[J]. Journal of Computer & System Sciences, 1975, 10(3):384-393.

[5] Cao Q, Wei Z B, Gong W M. An Optimized Algorithm for Task Scheduling Based on Activity Based Costing in Cloud Computing[C]// International Conference on Bioinformatics and Biomedical Engineering. IEEE, 2009:1-3.

[6] Xu H, Kang F, Li L. Task scheduling strategy based on multi fitness particle swarm optimization in cloud computing[J]. Icic Express Letters, 2014, 8(11):3165-3170.

[7] Jie X U, Zhu J C, Ke L U. Task Scheduling Algorithm Based on Dual Fitness Genetic Annealing Algorithm in Cloud Computing Environment[J]. Dianzi Keji Daxue Xuebao/journal of the University of Electronic Science & Technology of China, 2013, 42(6):900-904.

[8] Liu C Y, Zou C M, Wu P. A Task Scheduling Algorithm Based on Genetic Algorithm and Ant Colony Optimization in Cloud Computing[C]// International Symposium on Distributed Computing and Applications To Business, Engineering and Science. IEEE, 2014:68-72..

[9] Holland J H. Adoption in Natural and Artificial System[M]// Adaptation in natural and artificial systems. MIT Press, 1975:126–137.

[10] Srinivas M, Patnaik L M. Adaptive probabilities of crossover and mutation in genetic algorithms[J]. IEEE Transactions on Systems Man & Cybernetics, 1994, 24(4):656-667.

[11] Iosup A, Ostermann S, Yigitbasi N, et al. Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing[J]. IEEE Transactions on Parallel & Distributed Systems, 2011, 22(6):931-945.

[12] Ali S, Siegel H J, Maheswaran M, et al. Representing Task and Machine Heterogeneities for Heterogeneous[J]. Tamkang Journal of Science & Engineering, 2003, 3(3)

[13] Lipowski A, Lipowska D. Roulette-wheel selection via stochastic acceptance[J]. Physica A Statistical Mechanics & Its Applications, 2012, 391(6):2193-2196

Substitutability-Based Version Propagation to Manage the Evolution of Three-Level Component-Based Architectures

Alexandre Le Borgne[1], David Delahaye[2], Marianne Huchard[2], Christelle Urtado[1], and Sylvain Vauttier[1]

[1]LGI2P / Ecole des Mines d'Alès, Nîmes, France {Alexandre.Le-Borgne, Christelle.Urtado, Sylvain.Vauttier}@mines-ales.fr

[2]LIRMM / CNRS & Montpellier University, France {David.Delahaye, Marianne.Huchard}@lirmm.fr

Abstract

An important issue of software architecture evolution is the capability for architects to keep a trace of the evolution of their work. This paper states that existing research on versioning does not cope well with software architectures. Indeed, it does not propose any adapted solutions to manage the co-evolution of the different architecture representations produced during the development process. We base our work on a three-level architecture description language (ADL) named Dedal, which represents architectures at three abstraction levels. Moreover, Dedal provides a formal base for managing version propagation. It is based on component substitutability that generalizes Liskov's substitutability principle. We propose a set of rules to support the prediction of version compatibility in terms of impact on the different architecture levels.

Keywords: Component-Based Software Engineering, Architecture evolution, Architecture versioning, Component substitutability, Version propagation.

1 Introduction

One of the main issues of software evolution management is the versioning activity [11]. One needs to keep track of changes in software not only during software programming but all along its complete life-cycle, including early development stages such as specification or post-deployment phases such as maintenance. Moreover, maintaining the history of changes is not sufficient. Changes, and their side-effects, are to be managed in a very fine-grained manner in order to keep track of valid software configurations. Moreover, collaborative work accentuate the need for versioning systems.

With the growing complexity of software systems, versioning becomes a very essential activity either for developers who need to be able to reuse adequate versions of components, packages and libraries [20], or for users that need to maintain up-to-date versions of their applications.

Many version control mechanisms have been proposed over the past years. They track changes in either source code, objects or models [8]. However, despite the fact that software development gives an increasing importance to approaches based on software architectures [7], little works cope with architectural versioning issues such as keeping track of architectural decisions all along the life-cycle of software but also being capable of predicting compatibility of versioned architectural artifacts for reuse purpose and guiding architects and developers.

The remainder of this paper is organized as follows. Section 2 presents the background and motivations for this paper. Section 3 develops our contribution: a study on version propagation prediction based on component substitutability. Section 4 presents an overview of state-of-the-art work on the problematic of versioning. Finally, Section 5 concludes the paper with several perspectives.

2 Background and Motivations

In this section, we present a three-level architecture description language named Dedal [21, 15], which aims at supporting the main steps of software development. By nature, Dedal is an ideal candidate ADL to keep a record of software evolution during its whole life-cycle using versions. To do so properly, the impact of versioning a description on other levels is to be taken care of.

DOI reference number: 10.18293/SEKE2017-118

2.1 Dedal: A Three-Level Architecture Description Language

Dedal models architectures at three levels that correspond to the specification, implementation and deployment stages: (a) The **specification level** is composed of abstract component types. These types are called roles and describe the functionalities of the future software. (b) The **configuration level** is composed of concrete component classes that realize the roles. The relation between roles and component classes is a n to m relation. (c) The **assembly level** is a deployment model composed of component instances that instantiate the configuration. The syntax of Dedal is not detailed in this paper as the mechanisms that are presented here rely on general concepts. However, more information is accessible in one of our team's previous paper [22].

In Dedal, two properties guaranty the integrity of the three levels: (a) Intra-level consistency guarantees that all components are properly connected to each other. (b) Inter-level coherence guarantees that the configuration realizes all the roles that are defined in the specification and all the component classes from the configuration are instantiated in the assembly. However, those properties may be violated after changes and need to be recovered through a propagation mechanism within and / or between architecture levels. One of the major contributions of Dedal is its evolution manager which maintains the three architecture description levels coherent with one another. Dedal has been formalized [15] thanks to the B language [1]. This makes it possible to automatically calculate an evolution plan that, when it exists, restores the overall architecture coherence after any of its levels has been subject to change.

2.2 Motivation

Having multiple description levels makes versioning of component-based architectures at several abstraction levels necessary. Indeed, when one of the three architecture-description level evolves, it may have serious impacts on the other levels.

This issue has been discussed in a position paper [14] which uses a three-level versioning graph inspired from Conradi's taxonomy [8]. Authors differentiate two version types and also propose three strategies in order to manage three-level architecture versioning.

Version types are as follows: (a) **Revisions** are intended to replace their predecessor. This means that a revision should preferably be able to substitute to its previous version. A revision aims at improving an existing artifact. (b) **Variants** may coexist with other versions. A variant aims at adding new functionalities to an existing artifact.

Figure 1 illustrates versioning relations between three-level architecture descriptions. First of all, this example represents two versioning branches and gives an overview of how related description levels may be affected by single-level-versioning. The first three-level architecture version of this example is the A.1.0 version. Then we create two different versions that will fork the versioning graph into two branches. A.1.1 is a revision derivated from the assembly level revision itself. A.2.0 creates a new branch by creating a variant in specification level. Finally, either a variant or a revision of a single level may affect the other levels of the architecture. This is what it is shown in A.1.2 that propagates the revision of its configuration level *Config.1.1* to *Assembly.1.2* and A.2.0 where the variant of the specification level is propagated to the configuration and then to the assembly level.

A very interesting use of variant and revision concepts could be to predict the impact of changes in terms of retro-compatibility of versionned entities (*e.g.*, architectures, components etc.). Indeed, four types of derivation are identified: (i) **Compatible revision**, which means that the new revision does not raise compatibility issues (*e.g.*, bug fixes, refactoring) and does not imply to propagate changes. (ii) **Incompatible revision**, which means that the new revision raises compatibility issues (*e.g.*, new technology) and implies change propagation. (iii) **Compatible variant** provides an alternative version that does not imply to propagate any change in the architecture. (iv) **Incompatible variant** provides an alternative version that will require to propagate changes in the architecture.

Next section presents a study on version propagation in order to identify scenarios dealing with predicting the impact component substitution may have at any architecture level.

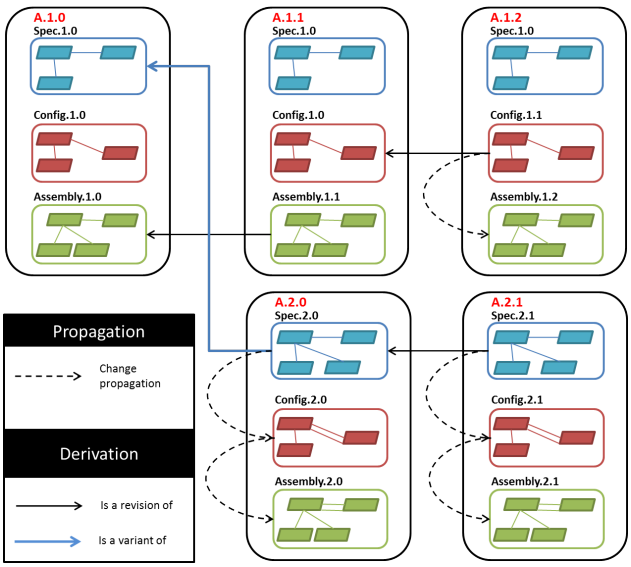


Figure 1. Versioning Architectures

3 Predicting Version Propagation

This section discusses a typology of the architecture evolutions managed by Dedal that aims at bringing semantics to versioning concepts of Dedal.

3.1 Typology of Architecture Evolution

The component substitutability relation has already been formalized in Dedal. This concept embodies the impact a change may have on architecture descriptions. This is the reason why this section introduces a typology of architecture evolution based on substitutable artifacts replacements instead of listing every single change operation on architecture artifacts. The aim of such a typology is to be able to identify which change is compatible or not with existing architectures.

Several change operations are relevant: (a) *Adding* new artifacts. (b) *Removing* artifacts. (c) *Replacing* artifacts with other that may be or not substitutable with the previous ones.

At architecture description level, a component may be replaced: (a) by a component that is *substitutable* for the replaced component. (b) by a component that is *not substitutable* for the replaced component.

Besides in a component-based architecture, several kinds of artifacts are subject to change: (a) components themselves, this is the most coarse-grained change, (b) a finer–grained change regards the interfaces of a component, (c) finally, the finest-grained change is performed on signatures.

When studying architecture versioning in a Dedal development, the initial level of change is especially relevant. Indeed, a very important aspect of having a three-level ADL is to be able to perform co-evolution of those levels according to the origin of the perturbation. This is discussed in next section.

3.2 Change Impact Analysis

As mentioned in the beginning of this paper, Dedal is a three-level ADL, which means that an initial change may occur at any of its architecture levels. This study is based on substitution of provided / required functionality signature. The notion of type is derived from Liskov *et al.* [13].

Notations. In order to avoid any kind of ambiguities, the used notation is described here.
$T_1 \prec T_2$: T_1 is a subtype of T_2.
$T_1 \preceq T_2$: T_1 is a subtype of T_2 or equal to T_2.
$T_1 \succ T_2$: T_1 is a supertype of T_2.
$T_1 \succeq T_2$: T_1 is a supertype of T_2 or equal to T_2.
$T_1 \parallel T_2$: T_1 is not comparable to T_2.
$(T_1 \not\preceq T_2) \Leftrightarrow \neg(T_1 \preceq T_2) \Leftrightarrow ((T_1 \succ T_2) \vee (T_1 \parallel T_2))$: T_1 is either a supertype of T_2 or not comparable to T_2.

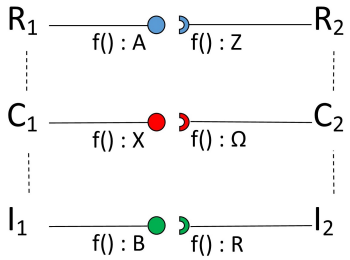


Figure 2. Base-Case: Functionality Connection Within a Three-Level Component-Based Architecture

$(T_1 \not\succeq T_2) \Leftrightarrow \neg(T_1 \succeq T_2) \Leftrightarrow ((T_1 \prec T_2) \vee (T_1 \parallel T_2))$: T_1 is either a subtype of T_2 or not comparable to T_2.
$T_2 \hookrightarrow T_1$: T_2 replaces T_1.

Functionality substitutable for another. For a provided functionality sp_{new}, being substitutable for another functionality sp_{old} means that (1) the return type of sp_{new} is equal or subtype [13] of the return type of sp_{old} and (2) that the input parameters of sp_{old} are subtypes of the ones of sp_{new} [3]. Conversely, for a required functionality sr_{new}, being substitutable for another functionality sr_{old} means that (1) the return type of sr_{new} is equal or a supertype of the return type of sr_{old},and (2) that the input parameters of sr_{old} are supertypes of the ones of sr_{new}.

Figure 2 is a base case that represents a specification composed of two component roles R_1 and R_2 that are realized respectively by the component classes C_1 and C_2, which are in turn instantiated by respectively I_1 and I_2.

3.2.1 Versioning at Specification Level

Table 1a summarizes what effect, replacing R_1 by its new version $R_1^{'}$ which provides a functionality $f() : Y$, may have on the configuration. Several outcomes are observable:

- **The version is not propagated.** This happens when the new version of the role still is compatible with other roles within specification, and the component class that realized the replaced role still is a subtype of the new role. The condition of non-propagation is summarized by $X \preceq Y \preceq Z$ for any replacement type. Y can either be substitutable for A or not.
- **The version is propagated.** This happens when information is lost during the replacement operation. According to the information that is lost, propagation may take different aspects: (i) **Inter-level propagation**, which occurs if Y is a subtype of X or if they are not comparable. (ii) **Intra-level propagation**, which is the result of breaking connections within the specification. This is possible if Y is a supertype of Z or

Hypothesis on types
$B \preceq X \preceq A \preceq Z \preceq \Omega \preceq R$
$Y \nrightarrow A$
Non-propagation
$X \preceq Y \preceq Z$
Propagation
Inter-level | Intra-level
$(Y \parallel X) \vee (Y \prec X)$ | $(Y \parallel Z) \vee (Y \succ Z)$
$(Y \parallel X) \wedge (Y \parallel Z)$

(a) Specification Level

Hypothesis on types
$B \preceq X \preceq A \preceq Z \preceq \Omega \preceq R$
$Y \nrightarrow X$
Non-propagation
$B \preceq Y \preceq A$
Propagation
Inter-level | Intra-level
$(Y \not\preceq A \Rightarrow \uparrow) \vee (Y \not\succeq B \Rightarrow \downarrow)$ | $Y \not\preceq \Omega$
$[(Y \not\preceq A) \vee (Y \not\succeq B)] \wedge (Y \not\preceq \Omega)$

(b) Configuration Level

Hypothesis on types
$B \preceq X \preceq A \preceq Z \preceq \Omega \preceq R$
$Y \nrightarrow B$
Non-propagation
$Y \preceq X$
Propagation
Inter-level | Intra-level
$Y \not\preceq X$ | $Y \not\preceq R$
$Y \not\preceq R$

(c) Assembly Level

Table 1. Replacing Components: Providing a Functionality

if they are not comparable. (iii) **Inter and intra-level propagation**, this is a combination of both propagation conditions. However the only reachable condition is that Y is not comparable neither to X nor Z.

3.2.2 Versioning at Configuration Level

Table 1b summarizes what effect, replacing C_1 by a third component class C_1', which provides a functionality $f() : Y$, may have on specification or/and assembly. Then several outcomes are observable:

- **The version is not propagated.** The condition of non-propagation is summarized by $B \preceq Y \preceq A$ for any type of replacement. Y can either be substitutable to X or not. $(Y \preceq A)$ ensures C_1' realizes R_1 and $(Y \succeq B)$ ensures I_1 can be used as an instance of C_3.
- **The version is propagated.** As configuration is the intermediate architecture level, then change may be propagated: (i) **To the specification (↑)** if Y is not a subtype of A. (ii) **To the assembly (↓)** if Y is not a supertype of B. (iii) **Within the configuration** if Y is not a subtype of Ω. This condition also implies at least a propagation to the specification since $(A \prec \Omega) \vdash (Y \not\preceq \Omega) \Rightarrow (Y \not\preceq A)$ (iv) **In every directions** with any combination of the previously expressed conditions. The change may be propagated in one, two or three directions at a time.

3.2.3 Versioning at Assembly Level

Table 1c is a summary of the impact that replacing I_1 by a third component instance I_1', which provides a functionality $f() : Y$, may have on the configuration. Two cases are possible:

- **The version is not propagated.** The condition of non-propagation is expressed by $Y \preceq X$ for any type of replacement (*substitutable* or *not-substitutable*). This condition ensures that I_1' instantiates C_1 and is compatible with I_2.
- **The version is propagated.** As previously, there exist several ways to propagate change: (i) **Inter-level propagation** if Y is not a subtype of X. (ii) **Intra-level propagation** if Y is not a subtype of R. This is also a sufficient condition for an inter-level propagation.

Tables 2a, 2b and 2c summarize symetric change impact analysis that corresponds to required functionality replacement at the three architecture levels (R_2, C_2 and I_2 are replaced by a component that requires a functionality $f() : Y$).

3.2.4 Generalization

1 to n replacement. The only cases that have yet been discussed are 1 to 1 replacement operations. However, this is sufficient to describe the propagation problem. Indeed, when a single role is realized by n component classes, then we can see those component classes as only one single composite component class that realizes a role. This is exactly the same situation when a single component realizes n component roles: we can see those component roles as a single one that exposes the interfaces which describe the n roles.

Multiple connections. A component interface may be connected to several interfaces in an architecture. A solution to generalize to such cases is to separately study each connection.

Thus the result of this study is as follows: substitutability is a good criterion for predicting impact on intra-level consistency. However a more detailed approach is needed for studying impact on inter-level coherence as it has been presented in the previously discussed tables.

4 Related Work

Initially, the versioning activity aimed at representing and retrieving the past states of a file during its evolution [10]. Versioning most often relies on text-based mech-

Hypothesis on types
$B \preceq X \preceq A \preceq Z \preceq \Omega \preceq R$
$Y \nrightarrow Z$
Non-propagation
$A \preceq Y \preceq \Omega$
Propagation
Inter-level | Intra-level
$Y \not\preceq \Omega$ | $Y \not\succeq A$
$(Y \parallel \Omega) \wedge (Y \parallel A)$

(a) Specification Level

Hypothesis on types
$B \preceq X \preceq A \preceq Z \preceq \Omega \preceq R$
$Y \nrightarrow \Omega$
Non-propagation
$Z \preceq Y \preceq R$
Propagation
Inter-level | Intra-level
$(Y \not\succeq Z \Rightarrow\uparrow) \vee (Y \not\preceq R \Rightarrow\downarrow)$ | $Y \not\succeq X$
$[(Y \not\succeq Z) \vee (Y \not\preceq R)] \wedge (Y \not\succeq X)$

(b) Configuration Level

Hypothesis on types
$B \preceq X \preceq A \preceq Z \preceq \Omega \preceq R$
$Y \nrightarrow R$
Non-propagation
$Y \succeq \Omega$
Propagation
Inter-level | Intra-level
$Y \not\succeq \Omega$ | $Y \not\succeq B$
$(Y \not\succeq \Omega) \wedge (Y \not\succeq B)$

(c) Assembly Level

Table 2. Replacing Components: Requiring a Functionality

anisms [5] as in version control systems like Git [19] or CVS [16]. However, models cannot be versioned as text.

4.1 Previous Work

In previous papers, Dedal has been introduced as an ADL for automatically managing evolution among multi-level component-based software architectures [15]. As changes may occur at any of the three architecture-levels of Dedal, the ADL has been formalized with the B language [1] to calculate evolution plans. An evolution plan aims at recovering consistence within architectural levels and coherence between those levels after a perturbation of any of the architecture levels. Also, the versioning activity has been considered in Dedal through a position paper, which presents a three-level versioning graph for managing Dedal architecture versioning [14].

4.2 Versioning Models

A model versioning process is composed of three steps [2]: (a) **The change detection phase.** Two types of approaches are identifiable for change detection [17]: (i) *State-based detection* only considers the final state of the modified versions. (ii) *Operation-based detection* relies on the model editor for keeping track of all the operations applied on the original version that lead to the final version. (b) **The conflict detection phase.** Conflicts may arise in case of parallel changes that are potentially overlapping or contradicting. For instance, several architects can modify a single version of a single model artifact simultaneously. (c) **The inconsistency detection phase.** Inconsistencies may happen while merging concurrent versions of a model artifact.

Models typically are entities linked to one another. Versioning one entity may thus have an impact on others. This makes co-evolution an interesting field of research. Research on model versioning has brought various approaches for managing co-evolution [17]: (a) **Inference approaches** [6] rely on meta-model comparison to generate a strategy for evolving models to conform to an updated meta-model. (b) **Operator approaches** [12] are based on patterns and are characterized by a set of predetermined strategies that can handle a step-by-step co-evolution of meta-models and models. (c) **Manual approaches** migrate models manually so they correspond to an updated meta–model.

Evolution in Dedal typically relies on inference mechanisms. The existing work on model co-evolution only copes with a top-down (meta-model to model) approach that corresponds to an historic use of meta-models in model-driven engineering (MDE). Indeed as a meta-model describes rules a model must respect in order to conform to it, there is no real need to adopt a bottom-up approach in MDE processes to evolve meta-models. Yet, multi-direction co-evolution is needed because an architect may need, for technical reasons, to adapt an implementation in such a way that it will not conform to the specification anymore. In such a case, change must be propagated from the implementation to the specification. This co-evolution need is well illustrated in Mokni *et al.* [15].

4.3 Versioning Architectures

Some existing work deals with versioning architectures. The few approaches that are presented here propose only basic mechanisms for architectural versioning that do not take into account the entire life-cyle of the software.

SOFA [4] does not completely support the whole life-cycle of software since it only provides two abstraction levels (configuration and non-descriptive assembly). Moreover, the finest-grained type SOFA takes into account is the interface type which is not enough to predict version propagation.

Mae [18] is based on xADL 2.0 [9] that provides two abstraction levels by distinguishing design-time and run-time. They thus do not take into account the whole life-cycle of software. The finest-grained type xADL 2.0 takes care of is the interface type and so it does not deal with input parameters, which is not enough for predicting version propagation. Mae enhances xADL 2.0 interfaces by adding interface elements that correspond to signatures and their input parameters but still does not cope with the entire software life-cycle.

5 Conclusion and Future Work

This paper introduces a study on version propagation, based on substitutability principles that have been formalized in Dedal. Through this study we could identify component substitution scenarios at any architecture-level that do not imply propagation of changes. As a consequence, we could also identify different propagation conditions within and / or between architecture levels. Then this study brings the capability of predicting the impact of new artifact versions (*e.g.*, architecture, component) by knowing their types. As a result of this study we could determine that component substitution is not a fine-grained enough criterion for predicting propagation on adjacent architecture levels, so we need to deal with parameter types into signatures. However, this is a sufficient criterion to predict propagation within an architecture level.

In future work, it will be essential to handle propagation problematics when adding or removing versioned artifacts in architectures in order to exhaustively predict version propagation.

References

[1] J.-R. Abrial. *The B Book - Assigning Programs to Meanings*. Cambridge University Press, Aug. 1996.

[2] K. Altmanninger, P. Brosch, G. Kappel, P. Langer, M. Seidl, K. Wieland, and M. Wimmer. Why model versioning research is needed!? an experience report. In *Proceedings of the MoDSE-MCCM Workshop@ MoDELS*, volume 9, 2009.

[3] G. Arévalo, N. Desnos, M. Huchard, C. Urtado, and S. Vauttier. Precalculating component interface compatibility using FCA. In J. Diatta, P. Eklund, and M. Liquière, editors, *Proceedings of the* 5^{th} *international conference on Concept Lattices and their Applications*, pages 241–252. CEUR Workshop Proceedings Vol. 331, Montpellier, France, Oct. 2007.

[4] T. Bures, P. Hnětynka, and F. Plášil. Sofa 2.0: Balancing advanced features in a hierarchical component model. In *Software Engineering Research, Management and Applications.* 4^{th} *International Conference on*, pages 40–48. IEEE, 2006.

[5] A. Cicchetti, F. Ciccozzi, and T. Leveque. A solution for concurrent versioning of metamodels and models. *Journal of Object Technology*, 11(3):1–32, 2012.

[6] A. Cicchetti, D. Di Ruscio, and A. Pierantonio. Managing dependent changes in coupled evolution. In *Proceedings of the International Conference on Theory and Practice of Model Transformations*, pages 35–51. Springer, 2009.

[7] P. Clements and M. Shaw. " The golden age of software architecture" revisited. *IEEE software*, 26(4), 2009.

[8] R. Conradi and B. Westfechtel. Version models for software configuration management. *ACM Computing Surveys*, 30(2):232–282, 1998.

[9] E. M. Dashofy, A. van der Hoek, and R. N. Taylor. A comprehensive approach for the development of modular software architecture description languages. *ACM Transactions on Software Engineering and Methodology*, 14(2):199–245, Apr. 2005.

[10] J. Estublier and R. Casallas. Three dimensional versioning. *Software Configuration Management*, pages 118–135, 1995.

[11] J. Estublier, D. Leblang, A. van der Hoek, R. Conradi, G. Clemm, W. Tichy, and D. Wiborg-Weber. Impact of software engineering research on the practice of software configuration management. *ACM Transactions on Software Engineering and Methodology*, 14(4):383–430, 2005.

[12] M. Herrmannsdoerfer. Operation-based versioning of metamodels with COPE. In *Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models, CVSM 2009*, pages 49–54, 2009.

[13] B. H. Liskov and J. M. Wing. A behavioral notion of subtyping. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 16(6):1811–1841, 1994.

[14] A. Mokni, M. Huchard, C. Urtado, , and S. Vauttier. A three-level versioning model for component-based software architectures. In *Proceedings of the* 11^{th} *International Conference on Software Engineering Advances*, pages 178 – 183, Roma, Italy, Aug. 2016.

[15] A. Mokni, C. Urtado, S. Vauttier, M. Huchard, and H. Y. Zhang. A formal approach for managing component-based architecture evolution. *Science of Computer Programming*, 127:24–49, 2016.

[16] T. Morse. CVS. *Linux Journal*, 1996(21es):3, 1996.

[17] R. F. Paige, N. Matragkas, and L. M. Rose. Evolving models in model-driven engineering: State-of-the-art and future challenges. *Journal of Systems and Software*, 111:272 – 280, 2016.

[18] R. Roshandel, A. van der Hoek, M. Mikic-Rakic, and N. Medvidovic. Mae—a system model and environment for managing architectural evolution. *ACM Transactions on Software Engineering and Methodology*, 13(2):240–276, 2004.

[19] L. Torvalds and J. Hamano. Git: Fast version control system. *URL http://git-scm. com*, 2010. last visited: 03.05.2017.

[20] C. Urtado and C. Oussalah. Complex entity versioning at two granularity levels. *Information systems*, 23(3-4):197–216, 1998.

[21] H. Y. Zhang, C. Urtado, and S. Vauttier. Architecture-centric component-based development needs a three-level ADL. In M. A. Babar and I. Gorton, editors, *Proceedings of the* 4^{th} *European Conference on Software Architecture*, volume 6285 of *LNCS*, pages 295–310, Copenhagen, Denmark, Aug. 2010. Springer.

[22] H. Y. Zhang, L. Zhang, C. Urtado, S. Vauttier, and M. Huchard. A three-level component model in component based software development. In *Proceedings of ACM SIGPLAN Notices*, volume 48, pages 70–79. ACM, 2012.

Towards a Models Traceability and Synchronization Approach of an Enterprise Architecture

Moreira, José Rogério Poggio
Department of Computing Science
Federal of University of Bahia
Salvador, Brasil
jose.poggio@ufba.br

Maciel, Rita Suzana Pitangueira
Department of Computing Science
Federal of University of Bahia
Salvador, Brasil
ritasuzana@dcc.ufba.br

***Abstract*- In the context of Enterprise Architecture (EA) modeling, the lack of alignment between the computational models constitutes an organizational problem. The root cause of this problem is the low traceability capacity and the lack of synchronization between the computational models present in the EA. Among the negative impacts related to this problem are, for example, the obsolescence of the models and the difficulty of carrying out analyzes of impacts and decision making in scenarios of organizational changes. Thus, in order to provide the alignment, understanding and adaptation of the corporate environment, from the institutional strategic level to the operational level of Information Technology (TI), focusing on the information systems, an approach is proposed to enable the traceability and synchronization between the computational models. The proposed approach in this paper consists of a: (i) meta-model set comprising the strategic, tactical and operational levels of the EA; (ii) traceability model that supports configuration and change management, through the use of COBIT and ITIL best practices; (iii) transformation process of models that, through the application of the Model Driven Development (MDD), aims to provide the synchronization between the elements of the different models present in an EA.**

Keywords- Enterprise Architecture; Model Synchronization; Traceability of Models; Alignment between Business and IT; Alignment between Strategy and Business Processes; Strategic Alignment; Business Model.

I. INTRODUCTION

An enterprise architecture is the organizing logic of strategy, business processes and IT infrastructure, including information systems, which reflects the integration and standardization requirements of an institution's strategic-operational model, such as the Zachman and TOGAF models that have driven and consolidated, respectively, the works in this area [10][19][22]. An enterprise architecture is formed by levels that are related: (i) strategic; (ii) business processes (tactical); (iii) services and; (iv) IT infrastructure (operational) [19]. In the modeling of enterprise architecture, the relationship between strategy, business processes and information systems can be represented by the alignment of the computational models elaborated within the organization [5]. The models represent instruments to reach and graphically represent the different levels of abstraction of enterprise architecture and to contribute to the achievement and maintenance of the strategic alignment between business and IT, in addition to their unions providing an integrated visualization of an organization's EA [19]. These models, then, represent the achievement of: (i) corporate and IT strategic planning; (ii) the business tactic to achieve the strategic objectives defined in the strategic planning and; (iii) the IT operation, focusing on information systems. However, the isolated construction of models such as the construction of a requirements model, without having a process model and a strategic model previously defined, can result in the elaboration of semantically fragile and non-aligned models with the business. Another relevant point is that if there is no defined strategy and the business processes are not established or are unstructured and without maturity, there is no reason for an institution to start developing its information systems.

In this scenario, to allow a better understanding of this research, this work defines the term alignment as the ability to trace and synchronize the strategic, tactical and operational structures present in the computational models of an organization.

The lack of traceability or its delivery in an inadequate way constitutes a problem and makes it difficult to see and understand how a set of models and their structures are related, which contributes negatively to the analysis of impacts in scenarios of organizational changes [2].

The absence of synchronization between different models (such as business process model and system requirements model) is another problem and results in the obsolescence of these models, which are now outdated and inconsistent, since the change in some element of the model, does not guarantee the updating of the other associated structures. This makes it difficult to maintain business strategy or at least generates a significant impact analysis effort in a scenario of change. In addition, there is still an equally significant probability of occurrence of failures, since the activity of checking the points that can be affected by a possible change, is manual [20].

In this way, it is proposed an approach that has the objective of enabling the synchronization and traceability between the computational models of an EA. The proposed approach consists of: (i) a meta-model set comprising the strategic, tactical and operational levels of the EA; (ii) an

DOI reference number: 10.18293/SEKE2017-021

independent traceability model that supports configuration and change management through the use of COBIT [8] and ITIL [15] best practices and; (iii) the application of Model-Driven Development (MDD), through the construction of a model transformation process, as a way to synchronize elements between different models.

The remainder of this paper is organized as follows: Section 2 presents the related works. Section 3 describes the meta-model set. Section 4 presents de traceability model. Section 5 describes model transformation process. Finally section 6 presents the conclusion of this work.

II. RELATED WORK

This work sought to gather relevant contributions, focused on the theme of strategic alignment between business and IT, focusing on the alignment between computational models of enterprise architecture. For this, a bibliographic survey was performed using systematic review techniques. Were considered the 1,500 works that were most relevant to the theme and that were returned by the search. Posteriorly, 96 works were identified from the association of their titles, with the respective theme of the research and the application of filters related to authors relevant to the theme. Of these, only 44 articles were selected, based on the analysis of inclusion and exclusion criteria defined for the research. The study covered the work that was identified during the research and which was published in the last 13 years (2004 to 2016) and was in English and Portuguese. The following databases were searched: (i) Compendex; (ii) Web of Science; (iii) IEEE Xplorer; (iv) Springer e; (v) Elsevier, in addition to the data sources: EJIS, SCIENCEDIRECT, WORLDSCIENTIFIC and AISEL. Also included in the bibliographic collection, from this work, sources of classic information [22] and reference [10]. From the analysis of the works it was possible to answer the following research questions:

- Q1. Are there techniques that align IT and business through traceability and synchronization between computing models at all levels of an enterprise architecture?
  A1. There was no identification of studies that performed alignment (traceability and synchronization) between computational models at all levels of EA. However, research was carried out that deals in a theoretical and partial way with the alignment between computational models [1][5][6][7][9].
- Q2. Is there a graphical language pattern to represent the elements present in the models of the different levels of enterprise architecture?
  A2. A graphical language pattern has not been identified for the modeling techniques that are used at the levels of an EA, since each level of the EA uses a different modeling technique and even within an EA level, such as the tactical, represented by business processes, exist different languages that can be used to modeling business process (BPMN, EPC, UML) [3][13][16].
- Q3. Are there standard concepts that are used to define the components used in the models of enterprise architecture?
  - A3. It was possible to identify a conceptual standard to define the components used in the models of enterprise architecture. At the process level, for example, there are process and activity concepts [13]. At the strategic and operational levels of IT, it was also possible to identify a conceptual standard [1] [5].

The analysis performed in the correlated work identified, showed that information technology is relevant to organizational performance and contributes directly to business operations. However, the lack of alignment between IT and business still prevails. For the analysis of the related works, tree technical criteria were defined. Each criterion represents the alignment between computational models at a certain level of the business architecture. The criterions defined were:

- Criterion I - Alignment between strategic models: that reflects the alignment between strategic corporate and IT maps;
- Criterion II - Alignment between strategic and tactical models: that reflects the alignment between strategic maps and business process models;
- Criterion III - Alignment between tactical and operational IT models: this reflects the alignment between business process models and IT requirements models, necessary for the development of information systems.

In order to improve the analysis of the works, tree attributes were defined for each criterion. Each attribute represents a characteristic that can be treated by an identified criterion. The attributes identified were:

- Theoretical: it should be used when the work theoretically addresses subjects related to the criterion;
- Traceability: should be used when the work deals with traceability between models;
- Synchronization: should be used when the work deals with synchronization between models.

Based on the identified criterion and attributes, the analysis of the studies found that the vast majority of research focuses on the theme related to criterion III. However, the study revealed that even in the theme of concentration (criterion III), the surveys do not cover the alignment in its completeness, because this does not contemplate the attribute of synchronization between models. The topics related to criterion I, as well as criterion II, have been very incipient, containing proposals that discuss their concepts and focus on the extraction of knowledge from strategic and tactical models (strategic maps and business process models) to identify requirements of systems.

The evaluation of the analyzed works indicated that the contributions of [2] and [5] that originated the languages BSC-P and ARMOR can be adapted to allow the closure of the gaps found in the themes associated to criteria I and II, since these works deal theorically traceability between models that are associated with criterion III. The B-SCP, also stands out as a contribution, due to the validation and verification of business requirements models, in terms of business process and strategy, which may allow an adaptation to the reality of the I criteria and II. [1], together with [7] also presented works

relevant to the topic of this article, when discussing concepts related to strategic maps, and can be used to support research in the first and second criterion.

The architecture model proposed by [6] also provides a relevant contribution and can be evaluated with a view to adapting to a more strategic business view. Considering the above, the evaluation validated the need to study the strategic alignment theme between business and IT, focusing on the computational models of an enterprise architecture, since it can be verified, the absence of work for some aspects (criterion I and II) and lack of completeness in others (criterion III).

Considering the above, the evaluation validated the need to study the strategic alignment theme between business and IT, focusing on the computational models of an enterprise architecture, since it can be seen, the absence of work for some aspects (criterion I and II) and lack of completeness in others (criterion III).

III. META-MODEL

The proposed set of meta-models aims to represent the semantics of EA, proposed by Zachman and TOGAF, and enable the tracking and synchronization of the models present in their different levels of abstraction, keeping the EA always aligned and without obsolescence. For this purpose, meta-models were constructed through the Meta Object Facility (MOF) [17], with the purpose of representing the main concepts, characteristics and relationships present in the strategic, tactical and operational levels of an EA.

Each meta-model is composed of: (i) concepts; (ii) aspects and; (iii) relationships. The concepts represent relevant elements that exist within the levels of the EA and are represented by meta-classes. The aspects represent the important characteristics that are related to the identified concepts and are represented by meta-attributes. Already the relationships represent the semantic associations that exist between the concepts present in the meta-models, being represented by associations. The meta-models were also designed to be independent of the technology used to construct the models (strategic, tactical and operational) that are associated with the respective meta-models. The concepts and aspects present in the set of meta-models were adopted from the perspective of techniques and good practices related to strategic planning, process management and software engineering, which, in turn, are associated with the strategic, tactical and operational levels of EA.

A relevant issue in the conception of this set of meta-models is the integrated and holistic approach to modeling (strategic, tactical and IT operational), viewing them as a single organizational model and reinforcing the institution's strategic thinking development (Objectives, strategies, strategic initiatives, processes and system requirements).

Figure 1 presents the proposal of the meta-model of strategic IT alignment, with a focus on information systems. In it, is possible to observe that there are three types of modeling: (i) strategic; (ii) tactic and; (iii) IT operational. These types of modeling are represented, respectively, in the set of meta-models by the packages: (i) MMEstrategic; (ii) MMTactical and; (iii) MMOperational.

To improve comprehension and understanding of the structures present in the meta-models was performed a description of all concepts and aspects present in the set of meta-models. However, this article presents only the detailing of the strategic meta-model, because it is the least addressed in the related works and most relevant within an institution.

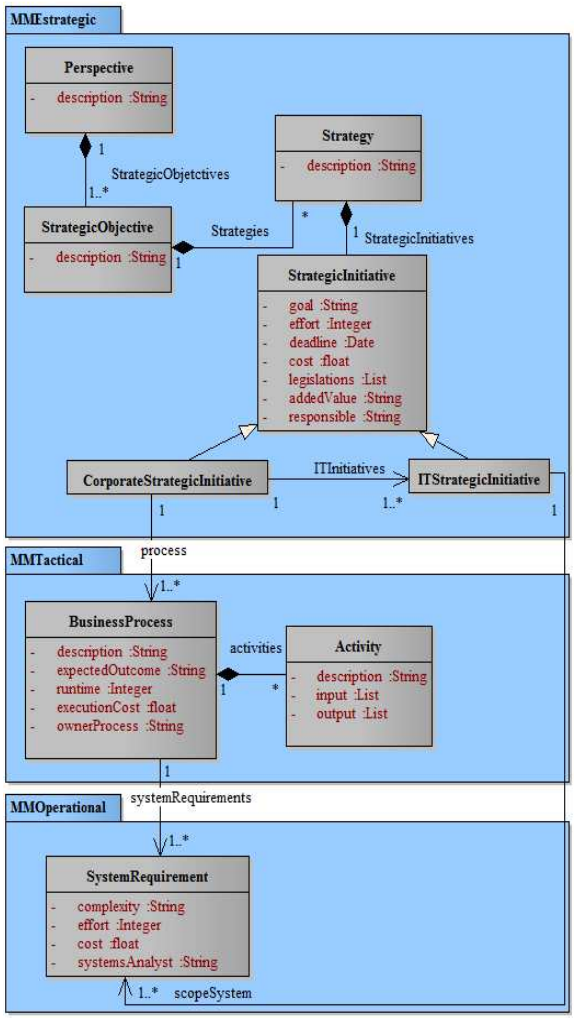


Figure 1. Meta-model Set.

- Concept: STRATEGIC INITIATIVE
  - Meaning: represents an action, project or program that must be executed to carry out the strategy that has been defined to achieve a strategic objective. A strategic initiative can be corporate or IT.
  - Component of the Meta-Model: meta-class "StrategicInitiative".
  - Associate EA level: strategic.
  - Good Practice of Origin: BSC and Grumbach [23] [24].
  - Aspects
    - Goal: represents the result that the organization wishes to achieve with the execution of the initiative. This aspect is represented by the meta-attribute "objective".

- Effort: represents the number of hours needed to carry out the initiative. This aspect is represented by the meta-attribute "effort".
- Deadline: represents the date set for the completion of the initiative. This aspect is represented by the meta-attribute "term".
- Cost: represents the financial importance associated with implementing the initiative. This aspect is represented by the meta-attribute "cost".
- Legislation: it represents the law, act or norm directly linked to the initiative and that must be observed during its planning and execution. This aspect is represented by the meta-attribute "legislations".
- Added Value: represents the benefit delivered to the organization by the realization of the initiative. This aspect is represented by the meta-attribute "addedValue".
- Responsible: represents the responsible role for the initiative. This aspect is represented by the meta-attribute "responsible".

- Concept: STRATEGY
  - Meaning: represents a direction that must be followed by the organization to achieve the defined strategic objective. A strategy can be corporate or IT.
  - Component of the Meta-model: meta-class "Strategy".
  - Associate EA level: strategic.
  - Good Practice of Origin: BSC and Grumbach [23] [24].
  - Aspects
    - Description: describes in detail the strategy that will be used for the organization to reach the strategic objective (corporate or IT). This aspect is represented by the meta-attribute "description".
- Concept: STRATEGIC OBJECTIVE
  - Meaning: represent a result that the organization wants to achieve, being critical for the institution to succeed in its area of operation. A strategic goal can be corporate or IT.
  - Component of the Meta-model: meta-class "StrategicObjective".
  - Associate EA level: strategic.
  - Good Practice of Origin: BSC and Grumbach [23] [24].
  - Aspects
    - Description: describes the expected result for the strategic objective defined during the strategic planning. This aspect is represented by the meta-attribute "description".
- Concept: PERSPECTIVE
  - Meaning: represents the organization's business perspective that, depending on strategic planning (Corporate or IT), is comprised of strategic corporate or IT objectives.
  - Component of the Meta-model: meta-class "Perspective".
  - Associate EA level: strategic.
  - Good Practice of Origin: BSC and Grumbach [23] [24].
  - Aspects
    - Description: describes the meaning of perspective for strategic planning. This aspect is represented by the meta-attribute "description".

IV. TRACEABILITY MODEL

This research works with the concept of horizontal and vertical traceability (forward and backward), and may also be associated with low and high granularity. The traceability model proposed in this work is based on several traceability models such as those proposed by [14][18][21].

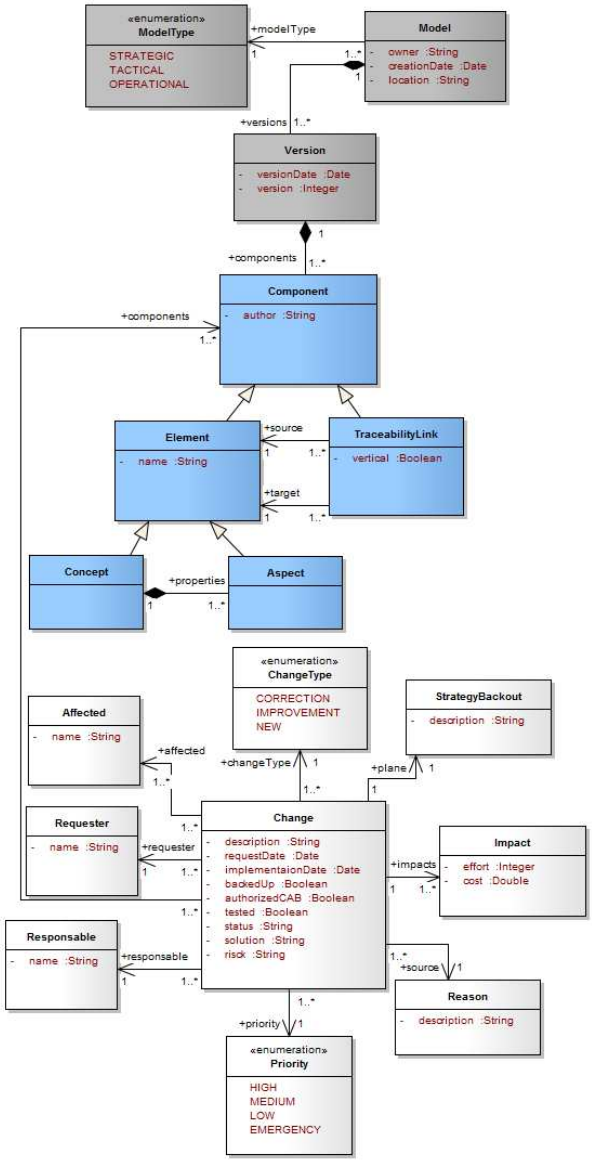


Figure 2. Traceability Model.

Figure 2 shows the traceability model proposed in this work. The model is divided into three parts. The first part (in blue) represents the structure that allows to trace the elements of the models, present in the different levels of the EA, of vertical or horizontal form, forwards or backwards and in the granularities high and low.

The first part is organized so that a component represents the abstraction of an element or a traceability link between two elements. An element may represent a concept or an aspect, and every traceability link is formed by two elements, one of origin and one of destination. In this way it is possible to create traceability links between concepts, between aspects and between concepts and aspects, besides enabling, as previously mentioned, traceability of low and high granularity, vertical and horizontal.

The second part (in gray) is formed by the configuration management structure. This structure is necessary to identify, control and provide the necessary information about an Organization Configuration Item (CI) [15]. A CI, in the context of this work, is represented by the Class "Model" that represents the strategic, tactical and operational computational models of enterprise architecture, along with its versions, types and components.

The third part (blank) is formed by the change management structure that allows changes to be recorded, prioritized, evaluated and authorized by the Change Advisory Board (CAB), tested and implemented, allowing control over the changes which occur on the models and their components, reducing the risk of incidents and consequently of damages to the organization.

Also according to the elaborated model, a change in a component can generate positive and / or negative impacts for the organization, besides having a requester, a responsible and the affected parties. A change may be requested for the purpose of correcting any problem, for the purpose of functional improvement or inclusion of a new component. In addition, a change always has a reason, which is why the change must be carried out, together with a strategy backout strategy, which is necessary to restore a component to the situation immediately prior to the change.

V. TRANSFORMATION PROCESS

In order to synchronize the elements of a set of models, preliminary theoretical studies were performed, demonstrating that the use of the Model-Driven Development (MDD) [12] approach would allow the synchronization of the concepts and aspects present in the models of an EA, allowing, therefore, that the elements of the models are always updated.

With MDD it is possible to transform more abstract models into more specific models. Among these models, there is a set of transformations rules that are applied to achieve the expected result.

The transformations are relevant and many works have been carried out in order to improve them and even automate those [4]. However, most of these works deal only with the transformations between the Platform Independent Model (PIM) and the Platform Specific Model (PSM), leaving behind the transformations that involve the Computation Independent Model (CIM) [12]. For [4], the transformations involving the CIM models are relevant, since it is through this model that an adequate understanding of the business is obtained that will give rise to the requirements that the information system must attend. Therefore, from the business specification represented by the CIM, the requirements of the information systems that make up the PIM must be determined. For this work, the CIM are represented by the strategic and tactical models of an EA. The operational model is a representation of the PIM.

The process aims to carry out transformations of models (strategic, tactical and operational), originated by the set of meta-models proposed in Section 3. The process, according to Figure 3, consists of five activities:

A. *Define Transformation Models*

- Description: This process activity aims identify and define the set of models that will be transformed, along with their respective meta-models, that should be used to carry out the transformations project.
- Input: Need to perform model transformation.
- Output: Model along with their respective meta-models.

B. *Define Transformation Strategies*

- Description: This activity aims to define the transformation strategy that consists of establishing the direction of transformation, which can be unidirectional or bidirectional, along with the type of transformation that may be out-place or in-place.
- Input: (i) model, along with their respective meta-models; (ii) the need to transform of the project of the models.
- Output: Transformation strategy.

C. *Design Transformations*

- Description: This activity aims to perform the specification of model transformations. This activity is responsible for the mapping of the transformations that will be performed, together with the definition of the transformation language that will be used to transform the models and the establishment of orchestration techniques of the model transformations that will be performed.
- Input: (i) model, along with their respective meta-models; (ii) the need to transform project models; (iii) transformation strategy.
- Output: (i) defined transformation language; (ii) transformation mapping; (iii) defined orchestration techniques.

D. *Implement Transformations*

- Description: This activity is responsible for choosing the most appropriate tool and implementation of transformations.
- Input: (i) model, along with their respective meta-models; (ii) the need to transform project models; (iii) transformation strategy; (iv) defined transformation

language; (v) transformation mapping performed; (vi) defined orchestration techniques.

- Output: defined tool and implemented transformations solution.

E. Execute Transformations

- Description: This activity refers to the execution, in practice, of the transformation of the models.
- Input: defined tool and implemented transformation solution.
- Output: transformed and synchronized models.

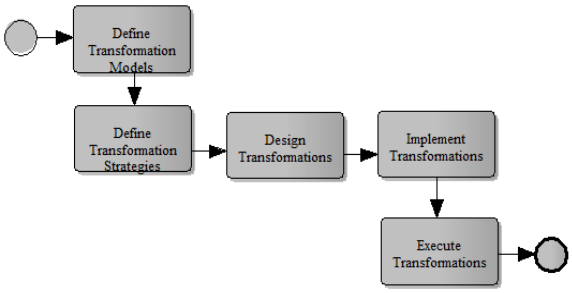


Figure 3. Model Transformation Process.

VI. CONCLUSIONS

This article presents the proposal that aims to enable the synchronization and traceability of the computational models present in enterprise architecture. The development of this work was carried out based on bibliographic research on the subject of enterprise architecture and under from the perspective of computational models and the strategic alignment of business and IT. At this stage of the work, no studies were found to perform the alignment (traceability and synchronization) of models in their completeness. The possible benefits expected for this research are:

- Creation of the integrated modeling (strategic, tactical and IT operational) approach in a traceable and synchronized way, viewing the models as a single organizational asset.
- Decreased obsolescence of organizational models and, consequently, of the organizational holistic view;
- Improve impact analysis in scenarios of organizational changes, allowing the institution to be more flexible;
- Facilitate the construction and maintenance of information systems.
- Contribute to the achievement of strategic alignment of IT, through by aligning the models present in A.E.

In this way, this work intends not only to improve the way organizations think about performing their computational modeling at the levels of enterprise architecture, but also to improve the way the current tools work with their models.

REFERENCES

[1]Babar, Abdul., Wong, B., Gill, A. Q., (2011), An Evaluation of the Goal-Oriented Approaches for Modelling Strategic Alignment Concept. Research Challenges in Information Science (RCIS), 1-8. Ieee.

[2]Bleistein, S. J., Cox, K., Verner, J., & Phalp, K. T., (2006), B-SCP: A requirements analysis framework for validating strategic alignment of organizational IT based on strategy, context, and process. In: Information and Software Technology.

[3]De Sousa, Rafael T.; et al, (2014), Business process modelling: A study case within the Brazilian Ministry of Planning, Budgeting and Management. In: CISTI.

[4]Dias, Felipe.; Morgado, Gisele.; Oscar, Pedro.; Silveira, Denis.; Alencar , Antonio Juarez.; Lima, Priscila. Schmitz, Eber. (2006). Uma Abordagem para a Transformação Automática do Modelo de Negócio em Modelo de Requisitos.

[5]Engelsman, W., Quartel, D., Jonkers, H., & Sinderen, M. V, 2011, Extending enterprise architecture modelling with business goals and requirements. Enterprise Information System.

[6]Fritscher, B., Pigneur, Y., (2011), Business IT Alignment from Business Model to Enterprise Architecture. Workshop on Business/IT Alignment.

[7]Giannoulis, C., Petit, M., Zdravkovic, J., (2011), Modeling Business Strategy: A metamodel of Strategy Maps and Balanced Scorecards. Research Challenges in Information Science.

[8]ITGI. (2012) COBIT 5. In: <http://www.isaca.org/cobit/Documents/COBIT-5-Introduction.pdf>. Accessed in June 2016.

[9]Jonkers, H., Lankhorst, M., Buuren, R. V., 2004, Concepts for modeling enterprise architectures. International Journal of Cooperative Information Systems.

[10]Josey, Andrew et al. (2011). TOGAF Version 9.1.

[11]Letelier, P. (2002). A framework for requirements traceability in UML–based projects. In International Workshop on Traceability In Emerging Forms of Software Engineering.

[12]Lucrédio, Daniel. (2009). Uma Abordagem Orientada a Modelos para Reutilização de Software. USP – São Carlos.

[13]Kurz, Matthias., (2016)., BPMN Model Interchange: The Quest for Interoperability., 8th S-BPM.

[14]Oglio, Pablo Dall. Silva, João Pablo Silva da. Crespo, Sergio. (2010). Um Modelo de Rastreabilidade com suporte ao Gerenciamento de Mudanças e Análise de Impacto.

[15]OGC. The Official Introduction to the ITIL Service Lifecycle. UK: TSO, 2007.

[16]OMG., 2015, UML 2.5 Specification. In: http://www.omg.org/spec/UML/2.5/PDF/.

[17]OMG., (2016), MOF 2.5.1 Specification. In: http://www.omg.org/spec/MOF/2.5.1/PDF. Accessed in July of the 2016.

[18]Ramesh, B., & Jarke, M. (2001). Toward reference models for requirements traceability. IEEE Transactions on Software Engineering.

[19]Ross, J. W., Weill, P., Robertson D. C., (2008), Arquitetura de TI como Estratégia Empresarial.

[20]Salgado, Carlos., Machado, Ricardo., Suzana, Rita., (2013), Using Process-level Use Case Diagrams to Infer the Business Motivation Model with a RUP-based Approach.

[21]Sayão, Miriam. Leite, Julio Cesar. Rastreabilidade de Requisitos. (2006). Revista de Informática Teórica e Aplicada.

[22]Zachman, J. a. (1987). A framework for information systems architecture. IBM Systems Journal, 26(3), 276-292.

[23]Brainstorming. Contruindo o Futuro - O Método Grumbach de Gestão Estratégica. 2012. In: <https://www.stm.jus.br/midias-planejamento-estrategico/artigos>. Accessed in June of the 2016.

[24]Kaplan, R. S. e Norton, D. P. - A Estratégia em Ação – Balanced Scorecard. Rio de Janeiro: Campus, 1997.

Infrastructure Based on Template Engines for Automatic Generation of Source Code for Self-adaptive Software Domain

Gabriele Salgado Benato[1] and Frank José Affonso[2]
Dept. of Statistics, Applied Mathematics and Computation
São Paulo State University – UNESP
PO Box 178, 13506-900, Rio Claro, SP, Brazil
[1]gabibenato@gmail.com, [2]frank@rc.unesp.br

Elisa Yumi Nakagawa
Dept. of Computer Systems
University of São Paulo – USP
PO Box 668, 13560-970, São Carlos, SP, Brazil
elisa@icmc.usp.br

***Abstract*—Self-adaptive Software (SaS), a special class of software system, constantly deals with some type of changes (i.e., structural and/or behavioral) to meet the user's new needs or autonomously react to modifications in its execution environment. Software adaptation, when manually performed, becomes an onerous and error-prone activity. Automated approaches have been adopted as a feasible alternative to overcome such adversities because they reduce the human involvement to a minimum. Based on this scenario, a Reference Architecture for SaS (RA4SaS) [1], [2] was designed in previous work. This architecture enables the adaptation of SaS at runtime by means of a controlled adaptation approach. In this sense, an automated process and a complete infrastructure to allow the phases of such process were also developed. This paper presents the design and implementation of a new module for automatic generation of source code for our architecture based on template engines [3]. In short, the main benefits of this module can be summarized in three items: (i) higher design flexibility, maintenance and evolution of SaS; (ii) introduction of new capabilities for automatic generation of source code; and (iii) performance improvement. To present the operation of this new module, a case study was conducted. As result, we have observed that our architecture has good perspective to efficiently contribute to the SaS area.**

***Keywords*-Automated Process, Reference Architecture, Self-adaptive Software, Template Engine.**

I. Introduction

Over recent years, a significant increase in the complexity of software systems and their computational environments has been observed. In general, such systems share functional, nonfunctional, physical, and virtual requirements. The human ability to manage systems becomes insufficient as their complexity increases. Moreover, involuntary injection of faults or uncertainties has often configured as one of the major causes of system failures, especially in the context of Self-adaptive Software (SaS). Software systems that can autonomously react to changes in their environment or modify its structure and/or behavior to meet the user's new needs are called SaS [4], [5].

Software adaptation, when manually performed, becomes an onerous and error-prone activity. Adversities, such as time, effort, money and the involuntary injection of uncertainties by developers, may be pointed as the main negative factors that have hampered the adaptation conducted by humans [6], [7]. To overcome such adversities, automated approaches have been adopted as a feasible alternative to maximize the speed of SaS implementation and, at the same time, minimize the developers' involvement [4], [8]. According to Salehie and Tahvildari [4], SaS development may be based on an external approach, which segments this type of software in two layers: (i) adaptable software, which represents software entities that will be adapted; and (ii) adaptation engine, which contains the adaptation logic. Regarding the second layer, control loops [9] have been used to ensure that the adaptation engine dynamically adjusts the adaptable software.

Based on this context, a Reference Architecture for SaS (RA4SaS) [1], [2] was designed in previous work. In short, this architecture is based on computational reflection [10] and an external adaptation approach [4], which enables the adaptation of software entities at runtime by means of a controlled adaptation modality. The software engineers define the adaptation levels of each software entity by means of annotations. Next, the adaptation of each entity is conducted by an automated process, since capabilities of human administration show decreasing relative effectiveness. For instance, some tasks have been difficult to manage, introducing potential problems, such as change management and simple human error.

Although our architecture (RA4SaS) has already been instantiated for a concrete architecture, the generation of source code, a crucial activity for the execution of this automated process, has shown some limitations. For instance, developer's interventions were required to meet different application domains, i.e., the software engineers were responsible for making adjustments to meet the requirements of an application domain, architectural models, among other requirements. In this sense, this paper presents the design and implementation of a new module for the RA4SaS based on Template Engine (TE). According to Bergen [3], TEs can be used in software development that needs automatically generation of source code according to specific purposes. Regarding the design and implementation of this new module, the main benefits are: (i) higher flexibility of design, maintenance and evolution of the software entities; (ii) introduction of new capabilities for automatic generation of source code; and (iii) performance improvement.

The paper is organized as follows: Section II presents the

DOI reference number: 10.18293/SEKE2017-147

background and related work; Section III provides a description of our architecture (RA4SaS); Section IV shows the design and implementation of an infrastructure for automatic generation of source code for the RA4SaS; Section V presents a case study to show the applicability of this new module; and Section VI summarizes the contributions and presents perspectives for further research.

II. Background and Related Work

This section presents the background (i.e., concepts and definitions on self-⋆ systems, reference architecture and TE) and related work on our paper.

Self-⋆ Systems. SaS has specific features in comparison to traditional ones because this type of software constantly deals with structural and/or behavioral changes at runtime. Some of them deal with management of complexity, reliability in handling unexpected conditions (e.g., failure), changing priorities and policies governing the goals, and context conditions (e.g., execution environment). The SaS development has boosted self-⋆ properties in general-purpose software systems, such as self-managing, self-configuring, self-organizing, self-protecting, self-healing, and so on. These properties allow systems to automatically react to the users' needs or to respond as soon as these systems meet execution environment changes [11], [12]. According to Silva and De Lemos [13], there is a set of goals to be achieved so that structural and behavioral modifications are performed in the SaS without affecting its execution states. For these authors, an adaptation plan is a feasible solution to define which procedures shall be adopted so that such changes are implemented.

Reference Architecture. According to Nakagawa et al. [14], Reference Architectures (RA) refer to a special type of software architecture that have become an important element to systematically reuse architectural knowledge. The main purpose of such architectures is to facilitate and guide [14]: (i) the design of concrete architectures for new systems; (ii) the extensions of systems of neighbor domains of a RA; (iii) the evolution of systems derived from a different RA; and (iv) the improvement in the standardization and interoperability of different systems. Considering their relevance for the software development, different domains have proposed RAs. For instance, service-oriented architectures such as IBM's foundation architecture [15] and architectures for software engineering environments [16] are some of RAs found in the literature. Other areas/domains have also proposed RAs, including self-⋆ software (e.g., RA4SaS [1]).

Template Engines. According to Bergen [3], "TEs are used in software development scenarios where it is necessary to automatically generate text and format it according to specific processing rules". This definition fits perfectly the purposes of our RA and its automated process of adaptation, which aims to manage the changes (i.e., user's new needs or modifications in the execution environment) without human intervention. The main reason for this type of TE is that our RA is based on reflection and, subsequently, it was instantiated in Java programming language. Thus, after a detailed analysis, the FreeMarker TE was chosen to be used in this work for the following reasons [3], [17]: (i) it is an open source software; (ii) it is a general-purpose template; (iii) it is faster than others; (iv) it provides facilities of use (e.g., JSON support, shared variables, template loaders, among others); (v) it has automated support in Java IDE (Integrated Development Environment); and (vi) its templates are not compiled to classes, i.e., a template can be loaded or reloaded during runtime without redeploying the application.

As related work, Cheng et al. [5] presented a study that aims to summarize the state-of-the-art on software engineering for SaS, identifying the main gaps and challenges. According to these authors, the Model-Driven Development (MDD) was pointed as a challenge for software adaptation, since a key issue of this approach is to keep the runtime models synchronized with system modifications. In this sense, the authors emphasized the importance of the code generators to maintain the integrity between source code and models. Silva and De Lemos [13] developed a framework for automatic generation of processes for SaS based on workflow, model-based approaches and artificial intelligence planning techniques. A specific contribution of this work is the usage of model-based technology for the generation of adaptation plans, since different planning techniques, according to the needs of the application domain, may be used. Daniele et al. [18] developed a Service Oriented Architecture (SOA)-based, platform-specific framework for context-aware mobile applications. This framework is composed of a methodology based on Model-Driven Architecture (MDA) principles that relies on SOA for context-aware mobile applications. Hallsteinsen et al. [19] developed a framework for the development of self-adapting applications in ubiquitous computing environments. This framework is based on MDD and it aims to facilitate the design and implementation of context-aware adaptive applications by the reuse of modeling artifacts and adaptive components. According to these initiatives, model-driven approaches and source code generators have been used in the development of self-adaptive applications in different domains. As our RA is based on metamodel for adapting software entities, only a source code generator based on TE was designed and it shall be presented in this paper.

III. Reference Architecture for SaS

Figure 1 shows the general representation of our RA, which is composed of four external modules and a core of adaptation [1]. This RA works with a controlled adaptation modality, i.e., the software engineer must insert annotations in each software entity so that the automatic mechanisms can identify their adaptability levels. In short, such levels contain parameters that determine where the new changes can be applied. Thus, when an entity is developed, an automatic mechanism performs a scan process to inspect if such annotations were correctly inserted. After a validation process, these entities can be stored in the entities repositories (execution environment) so that they may be invoked in future adaptations. Next, a brief description of this architecture is presented.

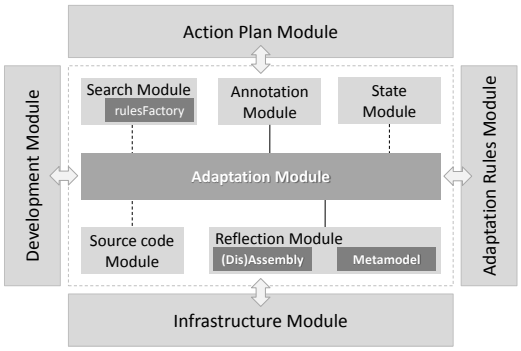


Figure 1. Reference architecture for SaS [1]

The "Development Module" provides a set of guidelines for requirement analysis, design, implementation, and adaptation of the software entities at runtime. The "Action Plan Module" aims to assist in the adaptation activity of such entities, providing controls for: (i) dynamic behavior, (ii) individual reasons, and (iii) execution state in relation to the environment. Thus, a framework [2] based on learning techniques [20] and the MAPE-K control loop [9], [21] and a tool called *DmS*-Modeler [22] were developed to assist the development of Decision-making Systems (DmS) for SaS, providing facilities for modeling, calibration of such system, and automatic generation of execution infrastructure. The "Adaptation Rules Module" provides a set of rules (i.e., metrics) for adapting software entities. The "Infrastructure Module" provides support for software entities adaptation at runtime, i.e., a set of mechanisms for the dynamic compiling and dynamic loading of software entities. The "Core of Adaptation" represents an automated process composed of a logic sequence of nine well-defined steps so that the adaptation of the software entities is conducted with no human intervention [1]. To do so, this core was organized in six modules: (i) search, (ii) annotation, (ii) state, (iv) source code, (v) reflection, and (vi) adaptation. In order to detail the adaptation modality of our RA, the reflection and source code module will be detailed in this paper.

The reflection module is organized into two submodules. The first assists in the "disassembly" and "assembly" of the software entities by using the annotation module for obtaining adaptation levels supported by each entity. Next, structural and behavioral information is recovered via reflection and inserted into a metamodel, which is inside the second submodule. New information can be either added or removed from this metamodel, according to adaptation interests. This new metamodel is then transferred to the source code module so that the new software entities are generated. Figure 2 shows the UML model of this metamodel, which enables the adaptation of object-oriented software systems or make use of the structure of classes as units of software systems, as it can notice the presence of classes (`Class`), attributes (`FieldClass`), methods (`MethodClass`), among other. In this metamodel, classes which have "`Annotation`" prefix are used to manage the annotations responsible for the adaptation levels of each entity. Similarly, classes with the "`Persistence`" prefix are used to manager the annotations responsible for the persistence of software entities. To delimit the scope of this metamodel in relation to the persistence of entities, an Object/Relational Mapping (ORM) framework was used. This framework enables us to develop persistent classes following natural object-oriented idioms as our metamodel. Thus, developers do not have to worry about the generation of storage structures, since such they are automatically generated by the framework. This strategy can be considered relevant for developers, since they can develop self-adaptive software entities in a notation to their expertise area.

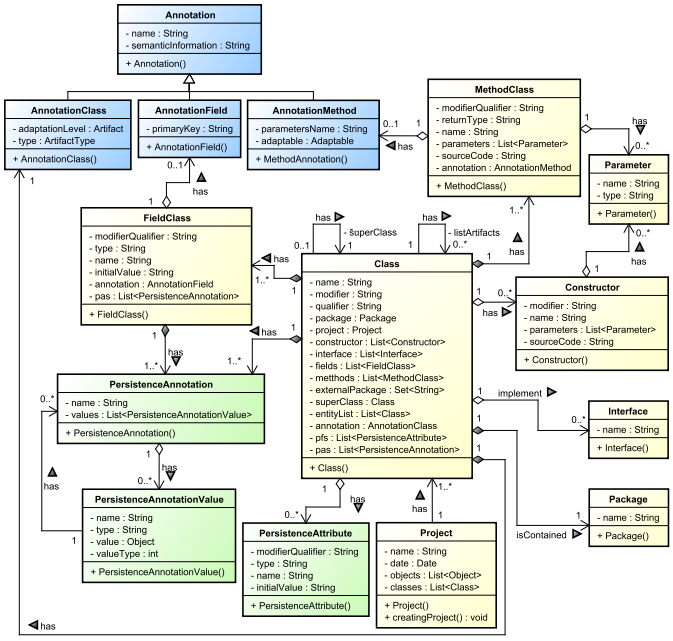


Figure 2. RA4SaS Metamodel for software entities

IV. Infrastructure for Automatic Generation of Source Code

The new generator module of source code is responsible to create the new software entities so that they can be compiled and inserted in the execution environment. Thus, the main purpose of this section is to present details on design and operation of such module in Section IV-A and a comparative analysis emphasizing the advantages of this module compared to the previous one in Section IV-B.

A. Design of the Generator Module of Source Code

Figure 3 provides an overview of the information flow contained in the generator module of source code, which is organized in three levels: (i) Metamodel, (ii) Template Engine, and (iii) Source Code. The first level represents the new metamodel with the adaptation changes (Figure 2). The second level contains all logic of this module, which must be able to generate the software entities according to the target domain and architectural models used in their development. Thus, software engineers and domain specialists must specify

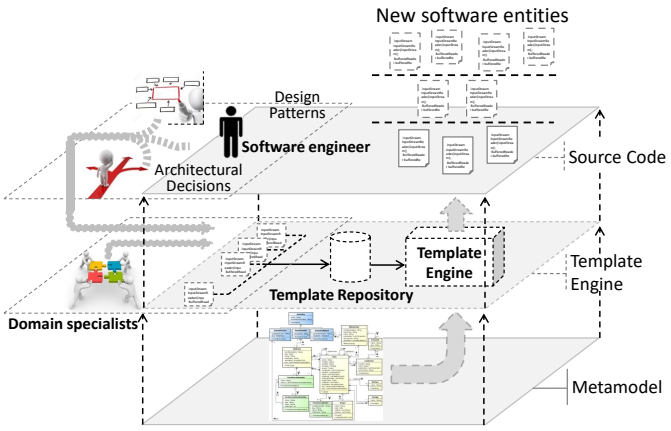


Figure 3. Source code module

design patterns, establish architectural decisions for the SaS development, and provide domain information to create a set of templates that will insert in the template repository so that the source code of each software entity can be generated by the "Template Engine" component. Design patters describe general solutions for design problems that occurs repeatedly in many projects including SaS domain. Architectural decisions represent implications for the environment in which the architecture will be deployed. Domain information encompasses the previous two, since the domain specialists must have a special knowledge to design each template according to requirements and restrictions and limitations imposed by the architectural and design models. Finally, the third level represents the source code for the new software entities.

The "Template Engine" component, illustrated on the second level, is a subsystem whose purpose is to generate source code for the software entities. Figure 4 illustrates the UML model of this subsystem, which is composed of three logical classes (`FileManager`, `FreeMarkerUtils`, and `TemplateManager`), an enum of constants (`FileType`), two external packages (`external`, and `freemarker.template`) and four stereotypes, i.e., two boundary classes (`Metamodel Boundary` and `Template Boundary`) and two control classes (`Metamodel Control` and `Template Control`). These packages contain a symbolic class (`Jacobe`) that represents the code formatter (`external` package) and three logical classes of the FreeMarker TE (`freemarker.template` package). The aforementioned stereotypes are used to represent: (i) the metamodels for the self-adaptive software entities, which are provided by the reflection module after insertion of adaptation changes; and (ii) the templates for generating software entities, since they may have been developed based on design patterns and architectural models; however, they may require a set of templates to generate the source code. Among of the logical classes, `TemplateManager` can be considered the main class of this subsystem because it is responsible for "orchestrating" activities so that the source code of software entities is generated. Initially, *Step 1*, the `start` method (`TemplateManager` class) receives as input a metamodel (from `Metamodel Control` and `Metamodel Boundary`) containing the software entities that will be generated. Next, *Step 2*, the `createEntity` method is invoked by the `start` method so that the software entities contained in this metamodel are fragmented and inserted in the entity list (`entities` attribute). In this step, the `FileType` enum is used, since provides a set of useful constants to maintain the nomenclature of such entities and to assist in the configuration of the code type that will be generated. Thus, for each retrieved entity, the `generateCode` method is invoked (*Step 3*). The purpose of this method is to match three important items: (i) nomenclature information (`FileType` enum), (ii) metamodel of each entity (`entities` attribute), and (iii) templates (from `Template Boundary` and `Template Control`). Regarding the operation, this method allows generating source code for three categories: logic classes, persistence classes, and business classes. These categories are identified by annotations inserted in the development phase (`AnnotationClass` – Figure 2). As *Step 4*, the `generateCode` method performs a call to `parser2Template` method of the `FreeMarkerUtils` class, transferring an entity map and template name that must be used to generate each entity. This method uses the templates provided by the software engineers, which are represented by stereotypes `Template Boundary` and `Template Control`. Moreover, the `FreeMarkerUtils` class can be considered as an abstraction for our subsystem because it encapsulates the classes of the FreeMarker TE (`freemarker.template` package). The source code files generated by the `FreeMarkerUtils` class are returned to `TemplateManager` class (`generateCode` method) so that they are formatted and inserted in their respective packages (*Step 5*). Operations for managing directories and files, both necessary in the source code generation process, are performed by the `FileManager` class. The `formatFile` method uses the `Jacobe` symbolic class to format the generated source code according to the standard established in the development phase. Finally, it is noteworthy that this process (Step 2 to 5) must be performed until the entity list (`entities` attribute – `TemplateManager` class) has been generated. Thus, the source code can be transferred to the compile module to be compiled and inserted into the execution environment.

B. Comparative Analysis

This section presents a brief comparative analysis between both modules (new and old). Next, a brief description of each feature is provided, as well as our analysis on both modules in relation to each of them, comparing their advantages and limitations.

Both modules have been developed based on the Java programming language. This language was selected because our RA is based on reflection [1]. Although the new module have been designed based on FreeMarker TE and uses the FreeMarker Template Language (FTL) for developing templates, the benefits highlighted in the next features are worthwhile. In addition, we can add the advantages already presented in Section II and the automated support by the main IDEs as complementary benefits.

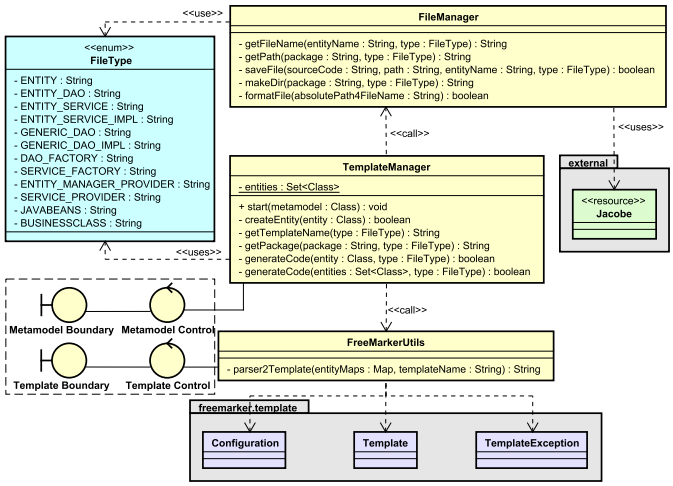


Figure 4. UML model of the source code module

Regarding the design, developers were obliged, during the development of the old module, to elaborate the "embedded templates" and processing logic (i.e., engine) of such templates in parallel. In this new version, the processing logic was abstracted to a simple class (`TemplateManager` class) and developers may devote their time in the preparation of code templates for software entities. Thus, a significant gain can be obtained when our RA is instantiated in a different domain that needs of changes for generating new entities.

As development approaches, two types were identified. The old module was developed based on internal approach and the new one based on external approach. As reported in previous feature, "embedded templates" and engine were developed by the developers in first approach (internal). This task was considered too costly (e.g., time and cost) to the developers when our RA had to be instantiated for a new application domain. On the other hand, a well-defined segmentation between code templates and engines was developed in the second approach (external). Thus, developers can coordinate their efforts in the development of code templates, since the engines of such templates were already implemented.

In the context of this paper, flexibility and maintainability can be considered related features because the first may assist or impair the second or vice-versa. According to previous paragraph, this new module has been more susceptible to change than the old one. In other words, when our RA is instantiated to meet a new domain, only the required entity templates should be developed. In addition, a significant minimization of code lines for processing templates in this new version compared to previous one must be highlighted. Therefore, this aspect can be considered highly favorable to maintain this new module.

V. Case Study

In this section, we present a case study to evaluate the applicability of the generator module of source code. The main purpose of this study is to demonstrate the real value of this module for our RA and, hence, for the SaS development. In previous work, we have instantiated our RA in Java and some validation types were conducted. Therefore, we will approach the same studies in this paper; however, emphasizing the use of the module presented in this paper. Next, a brief description of our subject application and the empirical strategies adopted for conducting this case study is addressed.

Subject Application. RA4SaS enables structural and behavioral adaptation at runtime [1]. Thus, for space reasons, only the structural adaptation was chosen for our empirical analysis, since this type of adaptation is sufficient to show the applicability of this module for the generation of source code. Thereby, a software entity will be used in the scope of this empirical study to meet two levels of granularity: (i) *association of entity*, which represents the addition of new information (i.e., entities) through the following relationships: aggregation, composition or association. In this paper, the composition relationship will be presented; and (ii) *extension of entity*, which represents the addition of new information (i.e., entities) by means of an inheritance relationship.

Empirical research strategy. Figure 5 illustrates the `Customer` software entity and the changes (i.e., association of entity and extension of entity) that will be applied to it. Before describing the case study, it is noteworthy that this entity belongs to the information system context for the bookstore management. Thus, to illustrate the first type of adaptation (i.e., *association of entity*), the original entity (`Customer`), initially developed to act in a local system, will be adapted to act in a web system with authentication. Based on this context, a `Login` entity with two attributes (`username` and `password`) must be created and added to the `Person` entity by means of a composition relationship. This entity (`Person`) receives the adaptation changes because it has an annotation (from "Annotation" module) that determines where such changes can be inserted. Then, a brief description of the adaptation process will be presented, but, for space reasons, the implementation commands are not shown in this paper.

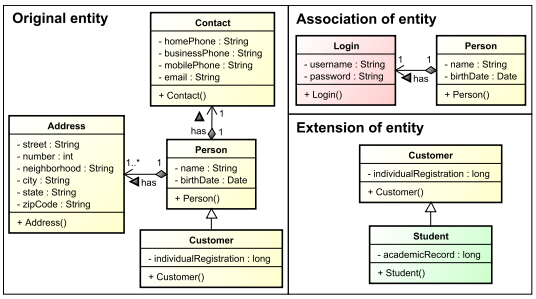


Figure 5. Adaptation types for the "Customer" software entity

To conduct the adaptation of the `Person` entity, the adaptation process of our RA will be used. In short, a metamodel (Figure 2) for this entity must be instantiated by the "Reflection" module so that the changes to be incorporated into it. Thus, a `Login` entity (i.e., an instance of `Class` – Figure 2) composed of two attributes (i.e., `username` and `password` – instance of `FieldClass`) is created in other metamodel (i.e., in a different plan). To associate these entities, the `Person`

metamodel must receive an attribute from `Login` metamodel, i.e., the entity list attribute (`entityList`) of the `Class` class. Metadata as type of relationship (e.g., composition), multiplicity (e.g., `ONE-to-ONE`), and navigability (e.g., bidirectional), also are provide in this step.

To show the *extension of entity*, it will be considered the modification of new `Customer` software entity. Now, this entity acts in a web system with authentication for bookstore management and it will be adapted to act in school management system. Similarly to the previous adaptation, the `Person` metamodel was instantiated so that a new entity was added via inheritance relationship. To do so, a metamodel for `Student` entity with `academicRecord` attribute must be created and associated with `Person` metamodel. Finally, specifically for this type of relationship, metadata determines the type of association (e.g., `EXTENDS`).

VI. Conclusions and Future Work

This paper presented the design and implementation of a module for automatic generation of source code for SaS. This module is part of a wider project, i.e., a RA4SaS that aims to support the development and adaptation of software entities at runtime [1]. Using this RA, software entities are transparently monitored and adapted at runtime without user's perception and developer's involvement. This RA uses a set of modules that coordinately work in an "assembly line", i.e., a software entity is automatically disassembled, adapted, and reassembled by the modules in an automated process. As discussed in Section IV, this new module aims to optimize the generation of source code and, at the same time, to boost the SaS development. Thus, the main contributions of this paper are: (i) SaS area, since we have a means that enables the development of self-adaptive software entities and it allows to adapt them at runtime; (ii) software architecture, because we have proposed the first RA for SaS based on reflection and, most importantly, we have worked in its evolution; (iii) software engineering community, since we believe that our RA may be adequately used together with software development processes that have been used by companies, since they seem to be complementary.

As future work, some activities are being planned: (i) the first is related to case studies that will be conducted for evaluation of the module presented in this paper, since the case study reported in Section V does not cover all cases and does not solve all problems related to software adaptation at runtime; however, other types of adaptation must be investigated; (ii) the second is related to the use of our RA in different domains, since we can get some indicators as flexibility and maintainability of this module; (iii) the third is related to execution performance of this module compared to previous one; and (iv) the fourth is related to the use of our RA in the industry, since we intend to evaluate the behavior of this module when it is applied in larger environments of development and execution. Finally, we have a positive scenario of research, intending to make this module (and our RA) become an effective contribution to the SaS community.

Acknowledgment

This research is supported by PROPe/UNESP and Brazilian funding agencies (FAPESP, CNPq and CAPES).

References

[1] F. J. Affonso and E. Y. Nakagawa, "A reference architecture based on reflection for self-adaptive software," in *SBCARS' 13*, 2013, pp. 129–138.

[2] 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*, 2015, pp. 1–6.

[3] J. V. Bergen, "Velocity or freemarker?" [*On-line*], *World Wide Web*, 2007, available in: http://www.javaworld.com/article/2077797/open-source-tools/velocity-or-freemarker.html, Accessed on March 08, 2017.

[4] 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.

[5] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. Di Marzo Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle, *Software Engineering for Self-Adaptive Systems: A Research Roadmap*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–26.

[6] F. J. Affonso and E. L. L. Rodrigues, "A proposal of reference architecture for the reconfigurable software development," in *SEKE' 2012*, San Francisco, USA, 2012, pp. 668–671.

[7] G. Coulson, G. Blair, and P. Grace, "On the performance of reflective systems software," in *PCCC' 2004*, 2004, pp. 763–769.

[8] F. J. Affonso, M. C. V. S. Carneiro, E. L. L. Rodrigues, and E. Y. Nakagawa, "A reference model as automated process for software adaptation at runtime," *IEEE Latin America Transactions*, vol. 13, no. 1, pp. 214–221, 2015.

[9] IBM, "An architectural blueprint for autonomic computing," [*On-line*], *World Wide Web*, 2005, available in: http://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf, Accessed on March 08, 2017).

[10] P. Maes, "Concepts and experiments in computational reflection," in *Object-oriented Programming Systems, Languages and Applications (OOPSLA' 87)*, New York, NY, USA, 1987, pp. 147–155.

[11] J. Kramer and J. Magee, "Self-managed systems: an architectural challenge," in *FOSE' 07*, 2007, pp. 259–268.

[12] D. Weyns, S. Malek, and J. Andersson, "Forms: a formal reference model for self-adaptation," in *ICAC' 2010*, 2010, pp. 205–214.

[13] C. E. Silva and R. De Lemos, "A framework for automatic generation of processes for self-adaptive software systems," *Informatica Journal*, vol. 35, no. 1, pp. 3–13, 2011.

[14] E. Y. Nakagawa, F. Oquendo, and M. Becker, "RAModel: A reference model of reference architectures," in *WICSA/ECSA' 2012*, 2012, pp. 297–301.

[15] R. High, S. Kinder, and S. Graham, "Ibm's soa foundation - an architectural introduction and overview," 2005, available in: http://signallake.com/innovation/soaNov05.pdf, Accessed on March 08, 2017.

[16] E. Y. Nakagawa, F. C. Ferrari, M. M. F. Sasaki, and J. C. Maldonado, "An aspect-oriented reference architecture for software engineering environments," *Journal of Systems and Software*, vol. 84, no. 10, pp. 1670–1684, 2011.

[17] A. FreeMarker, "What is apache freemarker?" [*On-line*], *World Wide Web*, 2016, available in: http://freemarker.incubator.apache.org/, Accessed on March 08, 2017.

[18] L. M. Daniele, E. Silva, L. F. Pires, and M. van Sinderen, *A SOA-Based Platform-Specific Framework for Context-Aware Mobile Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 25–37.

[19] S. Hallsteinsen, K. Geihs, N. Paspallis, F. Eliassen, G. Horn, J. Lorenzo, A. Mamelli, and G. Papadopoulos, "A development framework and methodology for self-adapting applications in ubiquitous computing environments," *Journal of Systems and Software*, vol. 85, no. 12, pp. 2840–2859, 2012.

[20] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing, 2005.

[21] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli, "A survey of autonomic communications," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 1, no. 2, pp. 223–259, 2006.

[22] F. J. Affonso, G. Leite, and E. Y. Nakagawa, "Dms-modeler: A tool for modeling decision-making systems for self-adaptive software domain," in *SEKE' 16*, 2016, pp. 617–622.

A Knowledge Engineering Process for the Development of Argumentation Schemes for Risk Management in Software Projects

Denise da Luz Siqueira[1], Lisandra M. Fontoura[1], Rafael H. Bordini[2], Luis A. L. Silva[1]
[1]Programa de Pós-Graduação em Ciência da Computação, UFSM. Santa Maria, RS, Brazil
{denise.siqueira, lisandramf, silva.luisalvaro}@gmail.com
[2]Programa de Pós-Graduação em Ciência da Computação, PUCRS. Porto Alegre, RS, Brazil
rafael.bordini@pucrs.br

Abstract – **The engagement of project stakeholders in collaborative debates of risk management has an important contribution to software projects. To promote the identification, (re)use and critical analysis of stakeholders' arguments in these debates, this paper lays out a knowledge engineering process for the development of "argumentation schemes" for risk management. This process covers activities of identification, interpretation and causal-and-effect analysis of typical risk statements. From such risk management information and reusing generalized argumentation templates from argumentation catalogues discussed in the field of Artificial Intelligence, the process leads to the specification, generalization, validation and indexing of the developed schemes. As implemented in our project, a web-based system to support the execution of these development activities allows the recording of these schemes in a semi-structured representation format. An argumentation scheme for risks of non-stable requirements is presented so as to show the reusable argumentation artifacts that can be produced when our development process is followed.**

Keywords-component: Risk Management; Argumentation Schemes; Argumentation.

I. INTRODUCTION

Risk management (RM) tasks act on the anticipation and treatment of risks that have a critical impact to the goals of a software project. As discussed in the CMMI standard [1], a more effective management of risks occurs when project stakeholders are engaged in the discussion of issues regarding the identification, analysis and response planning of risks. To support the (re)use of argumentation knowledge in the development of collaborative debates of RM, argumentation techniques have been exploited in the research project in which this work is situated [2-3] and in the solution of problems in other applications of Artificial Intelligence (AI) [4].

The collaborative debate of risks can be structured and mediated by argumentation-based communication protocols [5]. The exploitation of such protocols in a web-based Risk Discussion System [2-3] allows the capture and reuse of project stakeholders' arguments. The overall idea is to maintain these arguments in a knowledge repository formed of RM experiences. In this context, a common problem is that regular project participants (e.g. clients, managers and software engineers) tend to present risk analysis arguments in which relevant pieces of risk information may not be structured or even stated explicitly. In such debate situations, information to explain and justify the risks of a software project may not be presented and examined critically via users' arguments. To approach these issues, the argumentation literature presents generalized "argumentation scheme" specifications [6] so as to capture stereotypical patterns of presumptive reasoning. In contrast with these highly generalized argumentation templates, domain-specific scheme specifications are also presented as a form of better supporting debates in selected application problems (e.g. [7-8]). Despite the significance of scheme catalogues for the development of argumentation-based tasks of knowledge engineering, there are still many forms of argument to be identified as scheme specifications. On the assumption that losing any relevant argumentation knowledge in an application problem degrades the later usefulness on what is captured, the present work lays out the main knowledge engineering activities for the specification of semi-structured varieties of argumentation schemes for requirement risks in software projects. As a result of the proposed development process, novel schemes were not only presented in [9], but they were also assessed in collaborative debates in a real-life software project aiming the design and prototyping of a virtual tactical simulator (the SIS-ASTROS project).

The remainder of this paper is structured as follows. Section II reviews background information about RM and argumentation. Section III presents our scheme development process. Section IV discusses related work and Section V reviews our contribution and identifies future work.

II. RISK MANAGEMENT AND ARGUMENTATION

RM involves the discovery of events that can be a threat to the success of the software project so that a software development organization can plan risk response actions and build contingency reserves [10]. When analyzing failures in software projects, risks can be closely related to the poor management of requirements [10-12]. In this context, the engineering of requirements [13] is concerned with the assessment of clients' expectations, the evaluation and negotiation of a software solution, the validations of software specifications and

10.18293/SEKE2017-064

the management of client needs as they are transformed into software specifications. As requirement engineering is an initial software development task, requirement risks that ended being materialized in a project are likely to have a significant impact on the project (product) goals. As reviewed in [14], the majority of such requirement specifications are described in natural language or a semi-structured version of it, where just a small portion relies on formal specification languages.

The analysis of informal arguments relies on the identification of argument components. Approaches for argumentation diagramming [15-16] allow users to organize the components of an argument not only in terms of premises and conclusion, but also via detailed elements of the argumentation model due to Toulmin [17]. The overall idea is to promote users' critical thinking and improvement of users' argumentation skills by making argumentation elements explicit in semi-formal diagrammatic models, as implemented in the Araucaria system [15], for instance. In AI, argumentation frameworks [4] are usually investigated via some appropriate version of logic formalisms of reasoning with arguments. In these settings, argumentation schemes are developed as part of a standard knowledge acquisition and representation process (e.g. [18]). Although not represented as an explicit process model, the development of such schemes involves the modelling of these semi-formal or formal argument specifications (or both).

Argumentation schemes are intended to capture presumptive patterns of reasoning [6]. The underlying idea is that such schemes could support the expression of patterns of non-deductive reasoning, or even fallacies. In our research project, users' arguments presented in the debate of RM issues are captured and reused by taking advantage of the structure of these schemes. In doing so, one exploits an argumentation scheme as a template for knowledge acquisition in the collaborative RM problem. To help the understanding of the nature of a scheme, a long-established argument scheme specifying cause and effect arguments can be presented: *Major premise*: Generally, if A occurs, then B will (might) occur. *Minor premise*: In this case, A occurs (might occur). *Conclusion*: Therefore, in this case, B will (might) occur. The evaluation (or the validity) of scheme-based arguments presented in collaborative debates is an important argumentation characteristic. In the representation of argument schemes, this evaluation involves the identification and statement of questions regarded as critical. As presented in [6], "critical questions" (QCs) for the argumentation scheme from cause to effect are: How strong is the causal relation between (X) and (Y)? (If this causal generalization is true at all) Is the mentioned evidence (X) (if there is any) strong enough to warrant the cause-effect generalization as stated? Are there other factors (F) that would or will interfere with the production of the effect (E) in this case? Is (X) the main (or single) cause for the occurrence of (Y)?

Argumentation scheme specifications for users or systems to exploit and share scheme repositories are still been investigated. To approach this issue, argumentation mark-up languages have been proposed in the context of identification and visualization of argumentation schemes [19-20]. Those proposals are based on semantic web standards, which are often formalized on the basis of ontology representations. As described in [19], the Web Ontology Language (OWL) standard allows the construction of an ontology for the specification and annotation of arguments in different degrees of formalization, leading to a representation that can be processed by machines.

III. The Development of Argumentation Schemes for Collaborative Risk Management

To construct argumentation schemes not only to support the critical proposition and analysis of requirement risks, but also to promote the identification and specification of these kinds of risk-management arguments in a reusable argumentation template, a knowledge engineering process for scheme development should be put forward. Fig. 1 presents the UML activity diagram laying out our proposal for this process. To illustrate these activities, a new domain-oriented argumentation scheme developed in our project can be used: the "Argumentation scheme for risks of non-stable requirements". Such instability of requirements can be related to a volatile application domain, where clients either present new requirements in the course of a project or change requirement specifications made in the past (or both). In general, external factors of a software project can lead to new requirements or changes in the requirement specifications. To execute the development activities, a web-based system was implemented in our project (Fig. 2 - A). The activities of that process are as follows.

To select risk statements: it aims to compile a list of related requirement risk statements (here taken as arguments), to be structured as a reusable task-oriented argumentation scheme specification. To do so, the input artefacts are information repositories containing typical risk statements (or risk factors) [12, 21] in software projects. In addition to risk factors, concrete experiences of argumentation-based RM obtained in the past can be exploited in this activity. Moreover, development steps can be directed to the consultation with field experts in RM aiming to identify what risk factors are most subject to pro and con stakeholders' arguments. An example of such typical risk statements refers to arguments related to the proposition of risks about non-stable requirements. These kinds of requirement risk statements are the output artefact of this activity.

To describe risk interpretations: it aims to exploit risk interpretations in the understanding of the selected risk statements. Using these statements as input artefacts, this kind of risk argument can be augmented with the recording of their risk interpretations that are worth exploiting in user debates. The risk of requirements that are not stable, for example, can be contextualized in terms of constant changes in requirement specifications in a given software project. For this kind of risk, for instance, a possible interpretation can be stated as changes in the business domain while the project is running. Here, development steps are the consultation of available sources of risk interpretations, the discussion of such interpretations with field experts, and the collection of risk interpretation state-

ments. The output artefact of this activity is this list of risk interpretations along with their selected risk statement.

To analyze risk causes and effects: it aims to assess the causal factors that may lead to the materialization of the risk in a software project. To do so, a cause-and-effect diagram called Ishikawa diagram [22] can be used as illustrated in Fig. 2 (A). This diagram contributes to the understanding of the risk nature, and the consequent representation of a scheme that captures such kind of problem. The input artefacts here are the constructed list of risk interpretations along with their selected risk statements. In this cause-and-effect analysis, risk interpretations can be taken as risk causes in the Ishikawa diagram. Interpretations can also be analyzed according to different software development contexts such as project, client and business, for instance. Here, development steps are the consideration of the identified risk as an effect in this diagram, the identification of causal factors related to the materialization of this risk, and the representation of these pieces of information in the cause-and-effect diagram modelling and refinement. The cause-and-effect diagram constructed is the output artefact of this activity.

To reuse existing argumentation schemes: it aims to identify reusable templates in scheme catalogues in the construction of a task-oriented scheme specification. In our project, the generalized formulation of the "argumentation scheme from cause to effect" [6] was used in the specification of the cause-and-effect kinds of requirement risk arguments. Although our schemes were motivated by this cause-and-effect template, other requirement risk schemes may also be constructed from other generalized argumentation patterns. For example, risk assessment argumentation schemes could be specified to support the analysis of key stakeholders' claims as such arguments are understood via the "argumentation scheme from expert opinion" [6]. Here, development steps are the selection of argumentation scheme catalogues and consequent assessment of schemes available in them. In effect, this activity aims to find out generalized argumentation schemes, where their premises, conclusion and CQs are adjusted to reflect the debate needs of RM tasks. Here, the output artefacts are the formulations of generalized schemes to be reused in the specification of new argumentation templates for RM.

To represent scheme premises, conclusion and critical questions: it aims to deal with the concrete specification of the proposed argumentation scheme for RM. Here, the input artefacts are the selected risk statements, the risk interpretations identified, the results of the cause-and-effect analysis developed through the Ishikawa diagram and the generalized scheme specifications to be reused. In practice, scheme premises can be modelled from the causal risk statements analyzed in the cause-and-effect diagram. In effect, these premises can be described so as to capture risk interpretations. The conclusion of the proposed scheme is the very statement of the kind of risk that is the motivation for the scheme construction. Moreover, CQs can be formulated according to risk interpretations and the results of the cause-and-effect analysis. To structure a set of CQs, a knowledge engineer can exploit the argument nature of each question, checking if these questions capture exceptions to the structural rule of the scheme, for instance. In the "Argumentation scheme for risks of non-stable requirements", for example, a question asks whether the projects' application domain is really volatile, resulting in the constant changes in the specified requirements. Such questions show how debate users can attack the nature of the requirement risk arguments captured by this scheme, which is the true instability of the project requirements. In this activity, the output artefact is the concrete scheme specification, where this model is the object of additional steps of generalization, revision and validation.

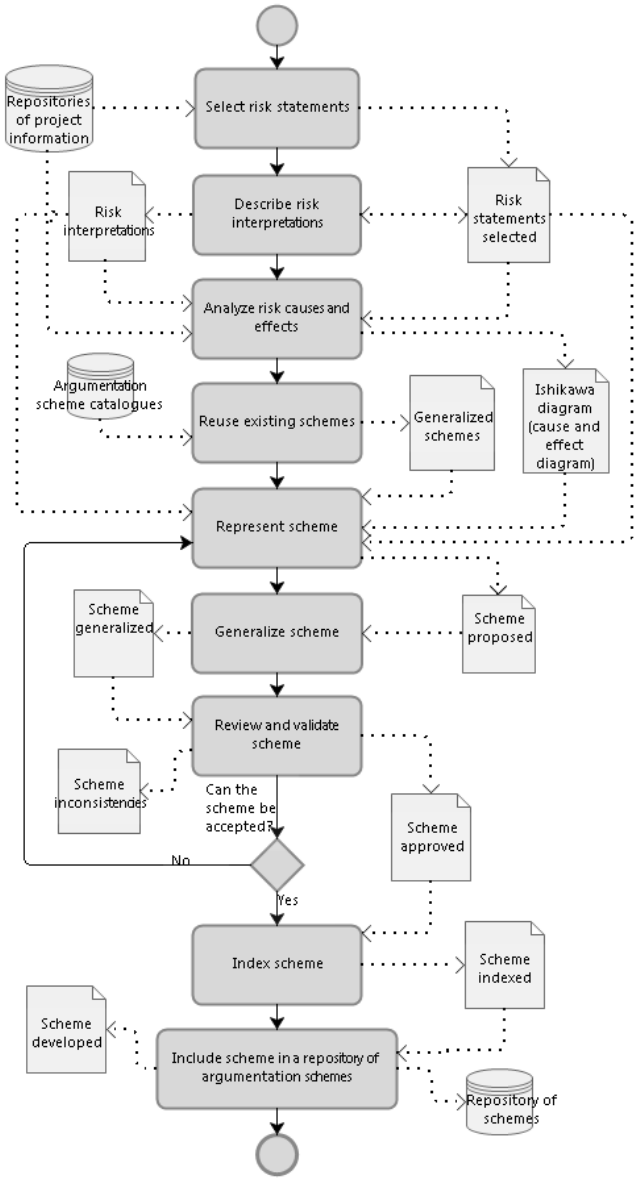


Figure 1. The process of argumentation-scheme development

To generalize the scheme specification: it aims to adapt the proposed scheme so that a more reusable task-oriented template specification is produced. Despite the general utility of generalized scheme formulations [6], risk-based schemes

ought to be specified in a language that regular stakeholders are more likely to cope with, promoting the understanding and reuse of these templates in collaborative debates. In addition to the rewriting of the terms and sentences used in the scheme description, this generalization activity is directed to the identification of scheme variables. These variables indicate places where these templates can be instantiated by debate participants. In debates, this is done when stakeholders use information from a current project situation in the construction of their scheme-based arguments. For instance, the first premise of the argumentation scheme for risks of non-stable requirements expresses that if the business domain is volatile, there will (might) have non-stable requirements. In the context of the collaborative debate of requirement risks in a particular project, this "business domain" term can be instantiated with the current project application domain. Here, the output artefact is the task-oriented scheme specification, which is now generalized to capture arguments for the RM problem.

To review and validate the scheme specification: it aims to evaluate the quality of the proposed scheme. To do so, a checklist is used in the revision and validation of this specification, which is the input artefact of this activity. In effect, this qualitative checklist should be adapted to knowledge engineers' needs so as to approach the construction of schemes for supporting the development of selected RM tasks. Items that are detailed in such checklists assess the underlying understanding of the argumentation scheme specification achieved, as well as the kinds of risk arguments that the scheme intends to capture. Other aspects can also be reviewed are: if the risk proposal is stated in an objective way in the scheme conclusion, if the vocabulary used to describe the scheme elements is usual in the problem scenarios where these schemes ought to be exploited, if key CQs are listed in the scheme representation and if the level of detail used in the scheme specification is appropriate for typical stakeholders. The output artefacts of this activity are the proposed scheme along with a list of possible inconsistencies identified in these revision and validation tasks. Once such a checklist is completed a scheme specification can be approved due to the fact that this specification shows selected characteristics of quality. As modelled in our project through the exploitation of this development process, a resulting argumentation scheme for risks of non-stable requirements is specified as:

Argumentation scheme for risks of non-stable requirements

Risk interpretations: changes in the business domain during the project development; large number of change requests regarding requirements coming from clients; lack of criteria for managing requirement changes in the project;

Major premise: If the business domain (D) is volatile, there will (might) have non-stable requirements (R)

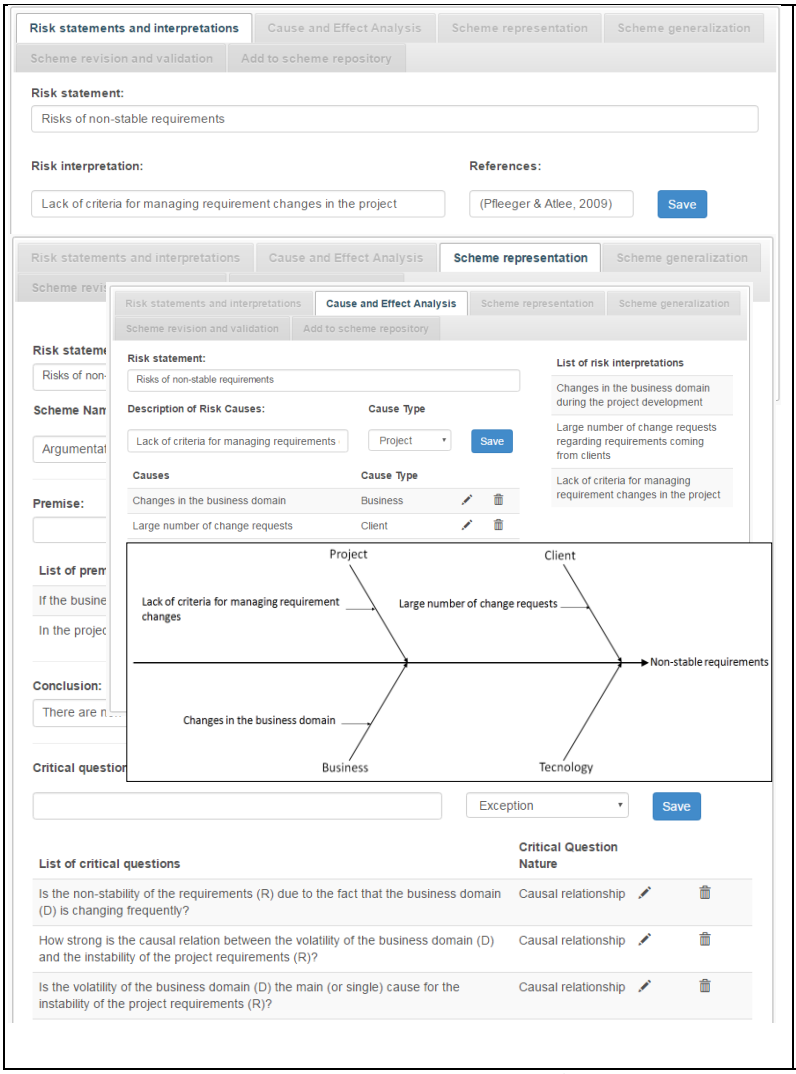
Minor premise: In project (P), the business domain (D) is volatile

Conclusion: There are non-stable requirements (R)

Critical questions: Is there evidence (X) that the high number of requests for changes in the requirement specifications is having a negative effect in project (P)? Are all requests for change (C) regarding requirement specifications (R) being accepted? Is the non-stability of requirements (R) due to the fact that the business domain (D) is changing frequently? To this scheme, the *CQs*, as in generalized formulations of argumentation schemes from cause to effect, are: How strong is the causal relation between the volatility of the business domain (D) and the instability of the project requirements (R)? Is the volatility of the business domain (D) the main (or single) cause for the instability of the project requirements (R)? Is there evidence (X) that the business domain (D) is volatile?

To the engineering of requirements, debate participants can question if the requirement engineering techniques are in place in the project, if there is a proper exploitation of these techniques, if the people involved in the development of requirement engineering tasks have the skills to adequately execute these tasks. The templates for these requirement engineering related *CQs* are: Are there requirement (elicitation, analysis, specification, validation, and management) techniques (T) to support (X) so that risk (R) is not in the project (P)? Are there requirement (elicitation, analysis, specification, validation, and management) techniques (T) being exploited properly by project stakeholders (K) so that risk (R) is not in the project (P)? Are the knowledge and experience of requirement engineers (E) adequate to do (T) so that risk (R) is not in the project (P)? In order words, it amounts to ask whether requirement engineers (E) have the right set of skills to develop the requirement engineering task (T). For instance, these requirement engineering questions in the context of the argumentation scheme for risks of non-stable requirements can be stated as: Are there requirement validation tasks (T) to support the exploitation of a volatile business domain (D) so that a risk of non-stable requirements is not in the project (P)? Are there requirement validation techniques (T) being exploited properly by project stakeholders (K) so that a risk of non-stable requirements is not in the project (P)? Are the knowledge and experience of requirement engineers (A) adequate to exploit a volatile business domain (D) so that a risk of non-stable requirements is not in the project (P)?

To index the scheme: it aims to organize the domain-oriented argumentation scheme proposed according to an indexing structure. This indexing consists of linking the scheme developed in a hierarchy of argument types. This argument-type concept aims to capture the nature of an argumentation scheme in an application problem, indicating what the scheme is about. Taking the approved argumentation scheme specification as input artefact, this specification is linked to a hierarchical list of argument types in this activity. For instance, the argumentation scheme for risks of non-stable requirements is an instance of *requirement risks*, which can also be understood as a type of argument for the *requirement risk management*. In essence, these indexing steps connect the new scheme specification to concepts used by users in the search of reusable schemes in argumentation repositories. Here, the output artefact is the indexed argumentation scheme.




```
<owl:Classrdf:ID="RequirementRiskScheme">
<rdfs:subClassOfrdf:resource="#ArgumentScheme"/>
...
<owl:Class rdf:ID=
"ArgumentationSchemeForRisksOfNon-StableRequirements">
<rdfs:subClassOfrdf:resource=
"#RequirementRiskManagementScheme"/>
</owl:Class>
<owl:ObjectProperty rdf:ID=
"supportsArgumentationSchemeForRisksOfNon-
StableRequirements">
<rdfs:domain>
<owl:Class>
<owl:unionOfrdf:parseType="Collection">
<owl:Class rdf:about=
"#ASI1Changes_in_the_business_domain_during_the_project_devel
opment(Pflrrger&Atlee(2009))"/>)"/>
...
<owl:Class
rdf:about="#ASP1If_the_business_domain_(D)_is_volatile,_there_wil
l_(might)_have_non-stable_requirements_(R)"/>
<owl:Class
rdf:about="#ASP2In_the_project_(P),_the_business_domain_(D)_is_
volatile"/>
<owl:Class rdf:about=
"#ASCThere_are_non-stable_requirements_(R)"/>
<owl:Class rdf:about=
"#ASCQ1Is_the_non-stability_of_the_requirements_(R)_due_
to_the_fact_that_the_business_domain_(D)_is_changing_frequently
?"/>
...
</owl:ObjectProperty>
<owl:ObjectPropertyrdf:ID="ArgumentSchemeHasVariables">
<rdfs:domain rdf:resource=
"# ArgumentationSchemeForRisksOfNon-StableRequirements"/>
<rdfs:range>
<owl:Class>
<owl:unionOfrdf:parseType="Collection">
<owl:Classrdf:about="#D_Business_Domain"/>
...
```

Figure 2. (A) A web-based system to support the execution of the scheme development activities proposed and (B) an excerpt of the argumentation scheme for risks of non-stable requirements represented in OWL

To include the scheme in a repository: it aims to construct a reusable argumentation repository, allowing the answer of queries and consequent reuse of developed argumentation schemes. The input artefact of this activity is the resulting scheme along with its indexing concepts. To compute inferences from scheme-based arguments stores in argumentation repositories, schemes need to be represented in computable formats of knowledge representation. As developed in our project, risk-related schemes are represented in the OWL format [23]. In this semi-structured model, scheme specifications can be queried by users and their computation tools. Fig. 2 (B) shows how the argumentation scheme for risks of non-stable requirements is represented in OWL. There, a class model is used in the capture of scheme interpretations, premises, conclusion and CQs. Scheme variables are also captured as classes, and relationships between scheme concepts are represented using class properties in this OWL model. In this final activity, the output artefact is the scheme developed which is represented in a semi-structured computational format.

IV. DISCUSSION

The definition of software development processes [24] allow stakeholders to standardize activities to be followed, the communication language to be used, and the techniques to be explored so that software development goals are achieved. To capture best practices, process models should not be immutable, since they need to evolve constantly to better support users in the solution of different problems. However, such improvements are possible provided that activities and artefacts that form these process structures are identified, allowing software engineers to evaluate and improve them continually. In this paper, we clearly identify a knowledge engineering process for the development of argumentation schemes for RM in software projects. Along with the discussion of each development activity identified there, the usefulness of this process is illustrated with the presentation of an argumentation scheme for risks of non-stable requirements (in addition to schemes presented in [9]), showing the reusable argumentation

artefacts that can be obtained when this development process is followed.

Most of the knowledge engineering work related to informal argumentation is focused on individual argument elements and their component parts that are identified in textual descriptions. This is usually developed through text-oriented annotation and diagramming resources adjusted to the analysis of argumentation concepts as implemented by the Araucaria system [15]. In this setting, generalized specifications of argumentation schemes [6] are also explored by users in the investigation of argumentation instances. As part of the development of critical argumentation skills, users are asked to recognize that an argument highlighted in such texts can be characterized by a given scheme. Although this line of argumentation work is relevant to the development activities proposed in this paper, these techniques are not directed or even organized as a knowledge engineering process for the specification of new instances of domain-specific argumentation schemes. Our process also considers the challenging linguistic and semantic analysis of large amount of textual information in specific domains, as shown by schemes for computer system safety engineering [7] and biological domains [8].

V. Concluding Remarks

Collaboration among project stakeholders in the engineering of requirements is crucial to achieve effective RM. The problem is that there is a significant gap between the information that is available in textual descriptions of RM users' arguments and the modelling and refinement steps resulting in generalized formulations of argumentation schemes.

In this paper, we approach the collection and structuring of well-formed stakeholders' arguments in collaborative debates of RM. In this context, this work contributes to the issue of laying out reusable argument templates to support these users in the construction of deeper analyzes of risks in their projects. In doing so, we detail a knowledge engineering process for the development of argumentation schemes for RM. We also describe a web-based system for supporting users on the development of the proposed scheme specification activities, aiming to facilitate the explicit representation of schemes in OWL. Preliminary evidence for the overall validity of this knowledge engineering process is demonstrated by a reusable set of argumentation schemes for the analysis of requirement risks in software projects as shown in [9].

Future work will involve attempting to exploit semi-structured approaches (e.g. OWL based) during the development scheme activities, rather than relying on informal scheme representations. We also plan to specify new scheme instances to be explored by project stakeholders in collaborative debates in different software project application domains. It amounts to involve both knowledge engineers and risk management experts to further experiment the proposed knowledge engineering process for the specification of schemes.

Acknowledgment

We thank the Brazilian Army for the financial support through the SIS-ASTROS Project (813782/2014), developed in the context of the PEE-ASTROS 2020.

References

[1] SEI, "CMMI® for Development, Version 1.3." p. 482.
[2] F. Severo, L. M. Fontoura, and L. A. L. Silva, "A Dialogue Game Approach to Collaborative Risk Management " in The 25th Int. Conf. on Software Engineering and Knowledge Engineering, Boston, MA, 2013, pp. 548-551.
[3] R. C. B. Pozzebon *et al.*, "Argumentation Schemes for the Reuse of Argumentation Information in Collaborative Risk Management," in Proc. of the 15th IEEE Int. Conf. on Information Reuse and Integration, Redwood City, CA, 2014, pp. 179-186.
[4] C. Chesñevar, A. Maguitman, and R. P. Loui, "Logical Models of Argument," *ACM Computing Surveys*, vol. 32, no. 4, pp. 337-383, 2000.
[5] P. Mcburney, and S. Parsons, "Dialogue Games for Agent Argumentation," in Argumentation in Artificial Intelligence, Boston, MA, 2009, pp. 261-280.
[6] D. Walton, C. Reed, and F. Macagno, *Argumentation Schemes*: Cambridge University Press, 2008.
[7] T. Yuan, and T. Kelly, "Argument Schemes in Computer System Safety Engineering," *Informal Logic*, vol. 31, no. 2, pp. 89-109, 2011.
[8] N. L. Green, "Identifying Argumentation Schemes in Genetics Research Articles," in Proc. of the 2nd Workshop on Argumentation Mining, Denver, Colorado, 2015, pp. 12–21.
[9] D. L. Siqueira *et al.*, "Argumentation Schemes for the Collaborative Debate of Requirement Risks in Software Projects," To Appear at The 29th Int. Conf. on Software Engineering & Knowledge Engineering (SEKE 2017), Pittsburgh, USA, 2017.
[10] PMI, "PMBOK Guide: A guide to the Project Management Body of Knowledge," 2013.
[11] K. Fenech, and C. De Raffaele, "Overcoming ICT Project Failures – a Practical Perspective," in World Congress on Computer and Information Technology (WCCIT), Sousse, Tunisia, 2013, pp. 1-6.
[12] M. Warkentin *et al.*, "Analysis of Systems Development Project Risks: An Integrative Framework," *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, vol. 40, no. 2, pp. 8-27, 2009.
[13] K. E. Wiegers, and J. Beatty, *Software Requirements*, 3ª ed.: Microsoft Press, 2013.
[14] E. Bagheri, and F. Ensan, "Consolidating Multiple Requirement Specifications through Argumentation," in ACM Symposium on Applied Computing (SAC '11), TaiChung, Taiwan, 2011, pp. 659-666.
[15] C. Reed, and G. Rowe, "Araucaria: Software for Argument Analysis, Diagramming and Representation," *Int. Journal on Artificial Intelligence Tools*, vol. 13, pp. 961-979, 2004.
[16] C. Reed, D. Walton, and F. Macagno, "Argument Diagramming in Logic, Law and Artificial Intelligence," *The Knowledge Engineering Review*, vol. 22, no. 1, pp. 87-109, 2007.
[17] S. E. Toulmin, *The Uses of Argument (Updated edition 2003)*, Cambridge: Cambridge University Press, 1958.
[18] A. T. G. Schreiber *et al.*, *Knowledge Engineering and Management - The CommonKADS Methodology*, Cambridge: The MIT Press, 2000.
[19] I. Rahwan *et al.*, "Representing and Classifying Arguments on the Semantic Web," *The Knowledge Engineering Review*, vol. 26, pp. 487-511, 2011.
[20] I. Rahwan, F. Zablith, and C. Reed, "Laying the foundations for a World Wide Argument Web," *Artificial Intelligence*, vol. 171, no. 10-15, pp. 897-921, 2007.
[21] B. Lawrence, K. Wiegers, and C. Ebert, "The top risk of requirements engineering," *Software, IEEE*, vol. 18, pp. 62-63, 2001.
[22] K. Ishikawa, *Introduction to Quality Control*: Springer, 1989.
[23] M. K. Smith, C. Welty, and D. McGuinness. "OWL Web Ontology Language Guide," https://www.w3.org/TR/owl-guide/.
[24] M. Hull *et al.*, "Software development processes — an assessment," *Information and Software Technology*, vol. 44, no. 1, pp. 1-12, 2002.

Argumentation Schemes for the Collaborative Debate of Requirement Risks in Software Projects

Denise da Luz Siqueira[1], Lisandra M. Fontoura[1], Rafael H. Bordini[2], Luis A. L. Silva[1]
[1]Programa de Pós-Graduação em Ciência da Computação, UFSM. Santa Maria, RS, Brazil
{denise.siqueira, lisandramf, silva.luisalvaro}@gmail.com
[2]Programa de Pós-Graduação em Ciência da Computação, PUCRS. Porto Alegre, RS, Brazil
rafael.bordini@pucrs.br

Abstract – **Managing risks in real-world software projects is of paramount importance. A significant class of such risks is related to the engineering of requirements, commonly involving the presentation and analysis of risk management arguments from both software engineers and clients involved in collaborative debates. In this work, drawing inspiration from argumentation theory in Artificial Intelligence, we introduce a number of "argumentation schemes" and associated "critical questions" to support such discussions. In doing so, we propose schemes related to risks due to excessive numbers of requirements; inadequate client representatives and poor understanding of client needs; incorrect, incomplete and conflicting requirements, and complex and non-traceable requirements. We also present a case study where the developed schemes were used to support the discussion of requirement risks in the context of a research and prototyping software project for the Brazilian Army.**

Keywords-component: Argumentation Schemes; Requirement Risks; Risk Management; Argumentation.

I. INTRODUCTION

Understanding and documenting key clients' needs is the foundation for success of software projects. As part of *requirement engineering* (RE) processes, these needs can be exploited through the development of RE tasks [1]. Despite these tasks, IT projects have changed in recent years because of increasingly complex requirements as described in [2]. Ubiquitous computing, inter-organizational systems, and an increment on consumer-targeted systems have created new challenges, making it difficult to manage requirements. This can be shown when major failures in software projects are analyzed [3]. It indicates that project stakeholders really ought to collaborate to the systematic identification of these problems in the initial phases of a software project, which is the time when the requirements are captured and analyzed as defined in software standards such as CMMI [4]. As explored in our research project [5-6], early consideration and treatment of software project risks amounts to the development of collaborative *risk management* (RM) tasks.

In collaborative scenarios, RM debates where "argumentation" techniques from Artificial Intelligence (AI) [7] can be used by project stakeholders to identify and analyze risks and their causes along with the proposition of risk response plans. As described in [8], argumentation processes can be used as a form of acquisition and construction of knowledge, which is a task based on presentation and computation of arguments and counter-arguments as well as the deliberations on particular issues. As described in [9], "argumentation schemes" allow the construction of such a dialogue-based system, where these schemes allow software engineers to describe argument structures commonly used in daily discourse. Such a scheme model presents a template for an argument type in an application problem, where this template can be used by stakeholders to elaborate and analyze argument instances to be advanced in their debates. As exploited in different applications [10-13], argumentation schemes can also be proposed so that they are used in the collaborative management of requirement risks in software projects.

As presented in [3], an open-ended questionnaire answered by IT professionals assessed specific factors contributing to either the success or the failure of software projects. Among the success factors identified, 2 of them refer to the management of requirements: the (high) level of abstraction for client participation and the thorough definition of requirements from the beginning. Among the failures identified, the study highlights the lack of requirement definition. From such results, this work proposes a new set of argumentation schemes from requirement risks. As the high-level of client participation impacts the success of a project, we specified schemes to capture arguments about requirement risks due to: the utilization of inadequate client representatives and requirements that do not fulfil clients' needs. The scheme for risks of inadequate client representatives assesses the experience of clients in the business domain of a project, as well as the interest and involvement of these clients in the elicitation of requirements. Moreover, the lack of involvement of clients can end up causing problems in the identification of their true needs. In effect, the argumentation scheme for requirements that not fulfil clients' needs is helpful in the assessment of the clients' commitment towards the elicitation and validation of requirements. A thorough initial definition of requirements can only be achieved when risks due to incorrect, incomplete and conflicting requirements are avoided. In this paper we introduce argumentation schemes to deal with such problems. In particular, the scheme for risks of incorrect requirements refers to whether the project scope is defined properly or not. The scheme for incomplete requirements aims to assess if all the requirement elicitation work has been done adequately so that the project could have better chances of achieving its goals [14]. In this context, issues regarding the specification of an excessive number of requirements can also be approached via

10.18293/SEKE2017-065

one of the schemes we put forward here. Conflicting requirements can occur when stakeholders have different views about project requirements or conflicting sources of information on the requirements. In general, the lack of requirement definition occurs when risks referring to the complexity of requirements are not prioritized. Among other reasons, complex requirements make it difficult for the project team to understand and share these requirements, reflecting how requirements are conceptualized and structured. According to [2], this kind of modelling issue indicates that it is difficult to understand, specify, and communicate the requirements. Finally, risks regarding the traceability of requirements ought to be examined when requirements are subject to many changes in the project, where this argumentation topic is considered in the specification of another scheme. To assess the usefulness of the schemes we created, we developed a collaborative debate on the management of requirement risks for a software project being carried out at UFSM. This project involves the research and prototyping of a virtual tactical simulator for the Brazilian Army – the SIS-ASTROS project. In this paper, a small fragment of this debate is used to illustrate how our requirement risk schemes were used by members of this project.

The remainder of this paper is structure as follows. Section II reviews background information about RM, RE and argumentation. Section III presents the new schemes developed. Section IV shows a fragment of a collaborative debate in which such schemes were used. Section V shortly discusses related works. Section VI summarizes the contribution of this paper and discusses future works.

II. Requirement Risks and Argumentation Schemes

The engineering of requirements in software projects is a process involving tasks for requirement elicitation, analysis, specification, validation and management [1]. In the requirement elicitation, clients' needs regarding their expected software products are described in a user language. In the requirement analysis, information captured during elicitation is gradually refined. This refinement leads to the construction of a model for the project requirements, where aspects of the function and behaviour of the software are identified. In the requirement specification, a conceptual model of the software is developed, which is used in the validation of the specified requirements and the planning of software development processes. In the requirement validation, the quality of the specifications of the requirement documents is examined by stakeholders. These specifications are reviewed to guarantee that the requirements were not captured as ambiguous descriptions, in addition to the detection of inconsistencies, omissions and errors. The team conducting such revision tasks should involve stakeholders interested in the examination of the specifications in the search for content and interpretation errors.

Despite all such efforts in RE, problems in those requirement specifications can be approached as risks in software projects. In simple terms, a risk is an event that can affect the goals of a project [15]. So, RM in software projects can be taken as the application of a set of principles and practices to the identification, analysis and treatment of risk factors aiming to prevent a project from failing, despite the fact that a risk can alternatively be treated as an opportunity in many software projects. This indicates that "risks that are due to software requirements" ought to be identified and their occurrence probability and project impact analyzed. Based on such risk evaluation tasks, risk treatment plans are established to risks with high priority in a project. In general, these RM processes demand a more intense communication process inside and outside the limits of the project. The need for collaboration is even higher when software projects involve stakeholders that are distributed geographically, in a scenario where the web is the primary communication medium. As described in [4], RM frameworks ought to be founded on significant steps of collaborative communication among stakeholders. Computational support for these activities is approached by web-based tools for recording and querying risk statements, their analysis and response plans. In effect, the collection of such RM information is naturally done in a dialogue where multiple stakeholders are involved, where collaborative steps support the discussion of different points of view and experiences. So, processes of collaborative decision making are what tend to make the RM most effective [4].

In argumentation scenarios involving project stakeholders, RM debates can ground the identification and analysis of risks and their response plans. As modelled by argumentation schemes, a set of speech acts is defined in the form of rules, which are used to give support to a conclusion statement advanced in such debates. In this case, the premises of an argumentation scheme allow users to capture common assumptions made in arguments. If such assumptions are accepted as true, this guarantees that the conclusion defined in the body of a scheme will also be accepted. In addition to a set of structural rules represented as premises and conclusion, each scheme is fundamentally defined by a set of *critical questions* (CQs). In case a user does not answer any one of these questions with relevant information in a debate situation, arguments that are based on the reasoning template of a scheme can be rejected. In general, the indication of variables helps the formulation of more generalized specification of argumentation templates, showing places where debate participants can shape scheme-based arguments with concrete pieces of information specific to the issue being discussed.

Argumentation schemes can be investigated to promote the identification of rather generalized patterns of argumentation. Described in an informal logic notation, the catalogue of such schemes [9] contains generalized templates for arguments from cause to effect, arguments from expert opinion, arguments from sign, etc. In fact, such kinds of templates could be exploited in the construction of tools to support collaborative debates. To approach such catalogues, it is relevant to notice that the analysis of risk causes and causal dependencies related to software failures is a significant RM characteristic (e.g. [16]). As defined in the "argumentation scheme from cause to effect" [9], argument templates can express relationships between causal factors and certain situations, where such situations are taken as effects from these causes. The generalized description of this scheme is: *Major premise*: Generally, if A occurs, then B will (might) occur. *Minor premise*: In this case, A occurred or will (might) occur. *Conclusion*: Therefore, in this case, B will (might) occur. *Critical questions*: How strong is the causal relation between (X) and (Y) (if this causal generali-

zation is true at all)? Is the evidence (X) mentioned (if there is any) strong enough to warrant the cause-effect generalization as stated? Are there other factors (F) that would or will interfere with the production of the effect (E) in this case? Is (X) the main (or single) cause for the occurrence of (Y)? Questions like these can be used to attack scheme-based instances of arguments when the debate participants present counter-arguments. In many situations, instances of questions created according to a scheme can present doubts about the structural connection between the premises and the conclusion of the arguments. From these questions, a scheme-based argument can be attacked in various ways, where the proposition of a counter-argument contains a conclusion opposing the conclusion of the original argument. Importantly, the assessment of such questions can also promote the presentation of supporting information to back up such arguments (Fig. 1 shows examples of these scheme-based speech moves).

Argumentation schemes are exploited in the solution of different application problems. In [10], an argumentation model proposed the use of schemes in the formalization of a dialogue protocol for agent communication. The protocol focuses on the presentation and computation of scheme-based arguments in favor of and against decisions, aiming to help users in the deliberation about the viability of organ transplants. In [11], schemes are used in the conciliation of results from similar biological experiments. Due to complexity of these experiments, results obtained when they are conducted may be conflicting or contradictory. In [12], a scheme-based approach to the representation and analysis of arguments regarding the security of computational systems is discussed. Those schemes were constructed from the generalization of safety-based issues commonly found in this domain. In [13], schemes are used in e-government and e-participation scenarios, where a web-based platform to promote democratic participation and debate of legal projects is proposed. Through a web-based tool, legal proposals can be input to the system, and citizens participate in the evaluation of those proposals through a website, either supporting or not the proposals. Schemes are used in the structuring of such proposals so as to promote their evaluation by members of the public. Despite the existence of these applications of schemes, and even though a preliminary set of schemes for RM [6] (as part of the research project where this work was conducted), the available schemes in the literature do not cover argument templates directed to the analysis of risks related to requirements in software projects.

III. Argumentation Schemes for the Collaborative Management of Requirement Risks

Following catalogues of requirement risks [17-20], argumentation schemes addressing the analysis of requirement risks can be proposed. To formulate these templates, requirement risks were analyzed according to the key RE tasks [1]. Taking advantage of a knowledge engineering process for the specification of argumentation schemes as proposed in [21], a set of 8 new schemes was structured via interpretations, premises, conclusion and CQs.

A. Argumentation scheme for risks related to excessive number of requirements

Risk interpretations: large numbers of requirement information sources (sometime different sources) available for query; large number of unnecessary requirements; too large and ambitious project scope; lack of consensus among project stakeholders;

Major premise: If there is a large number of information sources (S) available for consultation, there will (might) have an excessive number of requirements (R)

Minor premise: In the project (P) there is a large number of information sources (S) available for consultation

Conclusion: There is an excessive number of requirements (R)

Critical questions: Is the large number of available information sources (S) being questioned properly by project stakeholders (K)? Is there a too large and ambitious scope in the software project (P)? Is there a lack of consensus among project stakeholders (K) about the large number of requirements (R)?

For this scheme, an additional set of CQs that is typical in argumentation schemes from cause to effect are: How strong is the causal relation between the large number of information sources (S) and the excessive number of requirements (R)? Is the large number of information sources (S) the main (or single) cause for the occurrence of an excessive number of requirements (R)? Is there evidence (X) that there is a large number of information sources (S) available for query? Is there an excessive number of different information sources (S) available? As these cause-and-effect types of questions were instantiated in the context of the argumentation scheme from risks of excessive number of requirements, they can also be instantiated in the context of the other schemes presented later in this paper. Looking at RE tasks, it is also possible to ask if RE techniques are in place, if there is a proper exploitation of these techniques, if the people involved in the development of RE tasks have the skills to adequately execute these tasks in the project. Having a general applicability in all schemes proposed here, the overall templates for these RE kinds of CQs are: Are there requirement (elicitation, analysis, specification, validation, and management) techniques (T) to support (X) so that risk (R) does not apply to project (P)? Are there requirement (elicitation, analysis, specification, validation, management) techniques (T) being exploited properly by project stakeholders (K) so that risk (R) is not in the project (P)? Are the knowledge and experience of requirement engineers (E) adequate to do (T) so that risk (R) does not apply to project (P)? In order words, it amounts to question whether requirement engineers (E) have the right set of skills to develop the RE task (T). For instance, these RE questions in the context of the argumentation schemes from risks of excessive number of requirements can be written as: Are there requirement elicitation tasks (T) to support the exploitation of a large number of requirement information sources (S) available for query so that an excessive number of requirements (R) does not occur in project (P)? Are there requirement elicitation techniques (T) being exploited properly by project stakeholders (K) so that an excessive number of requirements (R) does not occur in project (P)? Are the knowledge and experience of requirement engineers (A) adequate to exploit a large number of requirement information sources (S) available for consultation so that

an excessive number of requirements (R) does not occur in project (P)?

The set of CQs defined here is not exhaustive. If users find it relevant to include new questions in this list, they should do so. A possible example for this is the formulation of questions related to the kinds of projects that are most commonly executed in a software development organization. In the SISASTROS project, CQs related to distributed software development issues and related requirement risks can be formulated. This is a context where the company acts on the design and implementation of simulation systems, which have particular risks associated.

B. Argumentation scheme for risks related to inadequacy of client representatives

Risk interpretations: client representatives not having experience in the target application domain; client representatives having opposing interests towards the project; client representatives are not truly committed to support the RE tasks;

Major premise: If client representatives (C) are not able to offer support to the RE tasks (T), there will (might) be client representatives (C) that are not adequate

Minor premise: In the project, the client representatives (C) are not able to offer support to the RE tasks (T)

Conclusion: The client representatives (C) are not adequate

Critical questions: Are client representatives (C) appearing to have just a vague knowledge about the target application domain (D)? Is there any reason for client representatives (C) to omit relevant requirement information (R)? Are client representatives (C) not really available to support the RE tasks (T)?

C. Argumentation scheme for risks related to incorrect requirements

Risk interpretations: misunderstanding the client needs; project scope not well defined; incorrect communication of project information by the client; lack of requirement validation tasks;

Major premise: If the client needs (N) are not being analyzed properly by requirement engineers (E), there will (might) be incorrect requirements (R)

Minor premise: In the project, the client needs (N) are not being analyzed properly by requirement engineers (E)

Conclusion: There are incorrect requirements (R)

Critical questions: Are the business rules (B) of the target application domain (D) too complex? Is there evidence (X) that the client needs (N) were not understood properly by requirement engineers (E), or that project information (I) was not properly communicated by clients? Are clients (C) really sure about what the significant requirements (R) are?

D. Argumentation scheme for risks related to complex requirements

Risk interpretations: large amount of information regarding project inputs and outputs; complex interaction with too many external interfaces (e.g. other systems, web services, etc); target application domain contains terms and concepts that are not well defined;

Major premise: If the target application domain (D) is complex for project stakeholders (K), there will (might) be complex requirements (R)

Minor premise: In the project, the target application domain (D) is complex for project stakeholders (K)

Conclusion: There are complex requirements (R)

Critical questions: Is there really a large amount of information related to inputs (I) and outputs (O) and interfaces (S) in project (P)? Is the complexity related to requirements (R) that are not being understood correctly? Are software artefacts (T) that help reduce the complexity of the requirement representations (R) being used?

E. Argumentation scheme for risks related to incomplete requirements

Risk interpretations: clients do not know what they want; clients forgetting to state important requirements; missing significant requirements as far as the project scope is concerned;

Major premise: If the requirements (R) have a large number of missing requirement specifications (S), there will (might) be incomplete requirements (R)

Minor premise: In project (P), requirements (R) have a large number of missing requirement specifications (S)

Conclusion: There are incomplete requirements (R)

Critical questions: Are clients (C) really sure about their needs in project (P)? Are there missing significant requirements (R) as far as the project scope (P) is concerned? Is there evidence (X) that clients are forgetting to state important requirements (R)?

F. Argumentation scheme for risks related to requirements that do not fulfil client needs

Risk interpretations: specified functionalities do not meet the clients' needs; requirement engineers not understanding the clients' needs; non-compliance issues between the specified requirements and the clients' needs for the project;

Major premise: If there are non-compliance issues (I) between the specified requirements (R) and the clients' needs (N), there will (might) be requirements (R) that do not fulfil clients needs (N)

Minor premise: In project (P), there are non-compliance issues (I) between the specified requirements (R) and the clients' needs (N)

Conclusion: There are requirements (R) that do not fulfil clients' needs (N)

Critical questions: Are clients (C) making their needs (N) clear (not vague or poorly visible) to requirement engineers (E)? Are client representatives (C) adequately supporting the development of the RE tasks (T)? Is there evidence (X) that client representatives (C) are committed to the development of the RE tasks (T)?

G. Argumentation scheme for risks related to conflicting requirements

Risk interpretations: conflicting information among specified requirements; different stakeholders proposing conflicting requirements; lack of a proper definition of scope for the project;

Major premise: If there are inconsistencies (I) among specified requirements (R), there will (might) be conflicting requirements (R)

Minor premise: In project (P), there are inconsistencies (I) among specified requirements (R)

Conclusion: There are conflicting requirements (R)

Critical questions: Are the conflicting requirements (R) related to divergences of opinion from different stakeholders (K)? Is there evidence (X) that there are conflicting requirements from different sources of information (S)? Are the conflicting requirements (R) due to incorrect understanding of clients needs (N)?

H. Argumentation scheme for risks related to non-traceable requirements

Risk interpretations: requirement sources are not tracked in the project; requirement changes are not managed in the project; lack of requirement management tasks;

Major premise: If the requirements (R) are not linked to their sources (S), there will (might) be non-traceable requirements (R)

Minor premise: In the project (P), the requirements (R) are not linked to their sources (S)

Conclusion: There are non-traceable requirements (R)

Critical questions: Are there requirement management techniques (T) being used by stakeholders (K) in the definition, capture and recording of links between requirement specifications (R) and their sources (S)? Are there requirement management techniques (T) being used by stakeholders (K) in the definition, capture and recording of links between requirement specifications (R) and requests for changes? Are the requirement changes (R) not being managed by stakeholders (K)?

IV. A Case for Argumentation Schemes in the Collaborative Management of Requirement Risks

A fragment of a collaborative debate of requirement risks in the SIS-ASTROS project (Fig. 1) can be presented to expose alternative forms that our schemes were used by participants of this project. First of all, this debate shows that a risk proposal can be grounded on a selected argumentation scheme (S next to a Propose_risk act of speech). The debate also shows that participants can use CQs when they want to refute a previous risk proposal argument. This kind of argument against the point being made can be advanced with the help of a selected CQ (Q next to an Argument_con). There, questions can also be submitted by participants, where a question is linked to a CQ of a selected scheme in the debate (Q next to an Ask). Alternatively, project information can be advanced as an answer to a CQ (Q linked to an Inform).

Propose_risk: The requirements can be incorrect @Manager01 **S**
- **Argument_con:** Some of these requirements were examined already by people from the army. They look like correct to me @Analyst01 ***Q***
  - **Argument_pro:** The project just started. We haven't developed a detailed specification of these requirements. This is the kind of specification to be better examined by them @Technical_leader01
- **Ask:** Don't you think that the army rules that should be used in this kind of simulation are quite complex? @Analyst02 ***Q***
  - **Inform:** We have a good collaboration with people from the army. Even if we consider this, I believe these rules are really complex for a software engineer with no military background @Manager01 ***Q***
- **Argument_pro:** There are military terms and abbreviations that will need to be worked out by the requirement engineers @Analyst03 ***Q***
- **Ask:** Should we consider that this project is dispersed geographically an issue for the understanding of the project requirements? @Manager02
  - **Inform:** This issue can be minimized if we have frequent meetings with people from the army. These meetings are even planned in the project plan @Manager01
- ...
- **Propose_plan:** We need to have meetings with the army people to reduce the impact of this risk in the project @Manager02
- ...

Propose_risk: There are complex distributed simulation requirements in this project @Technical_leader02 ***S***
- **Argument_pro:** This simulation architecture should consider the interaction between existing simulators in the army. It increases the complexity of the project @Manager02
- **Argument_pro:** As we know, some of these simulation systems are being developed in parallel projects to ours. All these simulators may end exchanging a lot of information if this distributed simulation architecture is not planned properly @Analyst02 **Q**
  - **Inform:** This integration of simulators will require a deeper investigation. That is good news since our results can greatly benefit the simulations that the army people want @Technical_leader01
- **Inform:** It should hire a military consulting company to help us with the specification and validation of these requirements @Manager01 ***Q***
- ...

Figure 1. An example of a requirement RM debate

Answers to a CQ can also be presented via an argument pro (Q next to Argument_pro), making either stronger or weaker the risk proposal analyzed by pros and cons. There, the overall debate that is promoted via argumentation schemes can help the gathering of project information from stakeholders. That is relevant in the determination of the probability and impact of a risk proposed, and their consequent prioritization. In the end, the answers presented to CQs used by debate participants can also contain pieces of explanation for the determination of different risk response plans to be used in the project. In the web-based system we have developed [5-6] that organizes these collaborative debates, the labels Q and S along arguments presented can be used by users. As a result, the debate system shows the schemes used in the construction of those arguments allowing users to reuse these templates in the construction of new arguments.

V. Discussion

Communication between project stakeholders is fundamental in the engineering of requirements [4]. Argumentation [7-8] focuses on the study and development of intelligent systems that model several kinds of communication aspects, such as how to mediate argumentation-based debates, for instance. Argumentation techniques are mostly directed to problems in the RE context such as the resolution of inconsis-

tencies in the consolidation of multiple requirement specifications [22] and the identification of security requirements in making mitigation decisions [23], for instance. In our work argumentation is used as a framework for the assessment of project risks related to requirement problems. In RM, these approaches for argumentation are being exploited as essential components of risk response plans, so as to assess the requirement specifications of a project in order to reduce the probability and impact of requirement risks. As possible forms of extending the approaches mentioned above, the argumentation schemes presented here focus on the capture of stakeholders' arguments in collaborative discussions of requirement risks. Through these schemes, the overall idea is to engage these stakeholders in the critical argumentation-based identification and analysis of these issues and consequent planning of risk responses.

Similar to [10-12], a domain-specific specification of argumentation schemes is exploited in the schemes proposed here. In [10], although the dialectical process modeled as the analysis of scheme-based pros and cons has a connection with how humans make qualitative decisions, the deliberation of solutions to RM problems is typically grounded on the additional utilization of more informal argumentation acts, where the dialogue used by project stakeholders is not limited to pros and cons kinds of scheme-based arguments. In [11], schemes are designed to support users in the analysis of the nature of inconsistencies in experimental results in Biology. In contrast with our work, a kind of cause-and-effect examination is not explicitly presented there as it is in our set of schemes. The schemes used in [13] rely on the generalized argumentation templates presented in [9]. Our work also exploits the reuse of such generalized scheme formulations, adapting a selected scheme from such catalogue to the specification of argumentation templates for the analysis of requirement risks.

VI. Concluding Remarks

One of the aims of this research is to reduce the communication gaps among project stakeholders particularly in relation to the collaborative development of RE tasks. In this context, the contributions of this paper are twofold. First, inspired by work on argumentation theory in the field of AI, we developed a number of new argumentation schemes to support collaborative discussions of requirement risks. Associated with these schemes (i.e., reasoning patterns) there is a large number of CQs which can be very useful to both software engineers and clients engage in such discussions. Second, we showed a case study conducted in the context of a real software project; the debate excerpt demonstrates how our schemes and particularly the CQs can be useful in practice. Future work includes the development of a larger set of requirement risks schemes, besides further experimentation with the tools we have developed [5-6] to support discussion of requirement risks based on argumentation schemes.

Acknowledgment

We thank the Brazilian Army for the financial support through the SIS-ASTROS Project (813782/2014), developed in the context of the PEE-ASTROS 2020.

References

[1] K. E. Wiegers, and J. Beatty, *Software Requirements*, 3ª ed.: Microsoft Press, 2013.

[2] L. Mathiassen, and T. Tuunanen, "Managing Requirements Risks in IT Projects," *IT Professional*, vol. 13, pp. 40-47, 2011.

[3] M. Warkentin *et al.*, "Analysis of Systems Development Project Risks: An Integrative Framework," *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, vol. 40, no. 2, pp. 8-27, 2009.

[4] SEI, "CMMI® for Development, Version 1.3." p. 482.

[5] F. Severo, L. M. Fontoura, and L. A. L. Silva, "A Dialogue Game Approach to Collaborative Risk Management " in The 25th Int. Conf. on Software Engineering and Knowledge Engineering, Boston, MA, 2013, pp. 548-551.

[6] R. C. B. Pozzebon *et al.*, "Argumentation Schemes for the Reuse of Argumentation Information in Collaborative Risk Management," in Proc. of the 15th IEEE Int. Conf. on Information Reuse and Integration, Redwood City, CA, 2014, pp. 179-186.

[7] C. Chesñevar, A. Maguitman, and R. P. Loui, "Logical Models of Argument," *ACM Computing Surveys*, vol. 32, no. 4, pp. 337-383, 2000.

[8] B. Moulin, H. Irandoust, and M. Bélanger, "Explanation and Argumentation Capabilities : Towards the Creation of More Persuasive Agents," *Artificial Intelligence Review*, pp. 169-222, 2002.

[9] D. Walton, C. Reed, and F. Macagno, *Argumentation Schemes*: Cambridge University Press, 2008.

[10] P. Tolchinsky *et al.*, "Deliberation Dialogues for Reasoning about Safety Critical Actions," *Autonomous Agents and Multi-Agent Systems*, vol. 25, no. 2, pp. 209-259, 2012.

[11] K. McLeod, G. Ferguson, and A. Burger, "Using Argumentation to Resolve Conflict in Biological Databases," *Proc. of Computational Models of Natural Argument (CMNA)*, vol. 9, pp. 15-23, 2009.

[12] T. Yuan, and T. Kelly, "Argument Schemes in Computer System Safety Engineering," *Informal Logic*, vol. 31, no. 2, pp. 89-109, 2011.

[13] D. Cartwright, and K. Atkinson, "Using Computational Argumentation to Support E-participation," *IEEE Intelligent Systems*, vol. 24, no. 5, pp. 42-52, 2009.

[14] PMI, "PMBOK Guide: A guide to the Project Management Body of Knowledge," 2013.

[15] B. W. Boehm, "Software Risk Management: Principles and Practices," *IEEE Software*, vol. 8, pp. 32-41, 1991.

[16] T. W. Kwan, and H. K. N. Leung, "A Risk Management Methodology for Project Risk Dependencies," *IEEE Transactions on Software Engineering*, vol. 37, no. 5, pp. 635 - 648, 2011.

[17] S. Amber, N. Shawoo, and S. Begum, "Determination of Risk During Requirement Engineering Process," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 3, pp. 358-364, 2012.

[18] S.-J. Huang, and W.-M. Han, "Exploring the Relationship between Software Project Duration and Risk Exposure: A Cluster Analysis," *Information & Management*, vol. 45, no. 3, pp. 175-182, 2008.

[19] L. Wallace, M. Keil, and A. Rai, "How Software Project Risk Affects Project Performance: An Investigation of the Dimensions of Risk and an Exploratory Model," *Decision Sciences*, vol. 35, pp. 289-321, 2004.

[20] L. Wallace, M. Keil, and A. Rai, "Understanding Software Project Risk: a Cluster Analysis," *Information and Management*, vol. 42, no. 1, pp. 115-125, 2004.

[21] D. L. Siqueira *et al.*, "A Knowledge Engineering Process for the Development of Argumentation Schemes for Risk Management in Software Projects," To Appear at The 29th Int. Conf. on Software Engineering & Knowledge Engineering, Pittsburgh, USA, 2017.

[22] E. Bagheri, and F. Ensan, "Consolidating Multiple Requirement Specifications through Argumentation," in ACM Symposium on Applied Computing (SAC '11), TaiChung, Taiwan, 2011, pp. 659-666.

[23] Y. Yu *et al.*, "Automated Analysis of Security Requirements through Risk-based Argumentation," *Journal of Systems and Software*, vol. 106, pp. 102–116, 2015.

An empirical study on software engineering and software startups: findings from cases in an innovation ecosystem

Leandro Pompermaier*, Rafael Chanin†, Afonso Sales‡, Kellen Fraga § and Rafael Prikladnicki¶

*† ‡ ¶Computer Science Department
§Business School
Pontifical Catholic University of Rio Grande do Sul (PUCRS)
Porto Alegre, RS, Brazil
*leandro.pompermaier@pucrs.br, †rafael.chanin@pucrs.br, ‡afonso.sales@pucrs.br,
§kellen.silva@pucrs.br, ¶rafaelp@pucrs.br

***Abstract*—Context:Software startups are increasingly seen as opportunities in a varieties of areas; from simple applications, as personal controls, to complex solutions, involving artificial intelligence or using big data. The process of developing these solutions happens quickly and in abbreviated form. Thus, software engineering should be adapted to better understand this world of software startups in a way to assist them in their customer discovery challenges and in the search for scalable business models. Main Goal: Understand the factors that positively and negatively influence the software development of software startups and present solutions found in an empirical study on startups located in an innovation ecosystem Methodology: We conducted interviews and observations as data collection techniques in eight software startups located at a Tech Park.**

***Keywords*—Software Startups; Software Engineering.**

I. Introduction

The technological evolution presented in the last years presents a significant challenge for the companies in general since they concentrate efforts to find new and more efficient ways of conducting their businesses [8]. There is a growing market for systems that operate at low cost, high quality and have a short development cycle. It suggests that the development approach using software engineering practices start on the need to achieve high quality and productivity.

In software development, some of the key factors affecting quality and productivity include people and procedures as well as technologies being employed to accomplish the intended activity [2]. Currently, there is a growing number of startups that develop innovative solutions. Startup could be defined as a human institution designed to deliver a new product or service under conditions of extreme uncertainty [10].We can define a subset of startups that have their software-based solutions as software startups or digital startups. These are considered to be newly created companies with no operational history and are incredibly fast in the production of cutting-edge technologies [9].

DOI reference number: 10.18293/SEKE2017-115

Innovation environments, such as universities, play a fundamental role in technological innovation and socio-economic growth in a region or country. According to Etzkowitz [6], the transformations that universities have undergone in recent years, the so-called University revolutions, have unified the three top strands of these institutions: teaching, research and economic and social development of the country. Business incubators are mechanisms of innovation and strategic process in the economic development of countries, states, cities and companies. The aim of incubators is to produce successful enterprises, which is why incubators help ventures to survive and grow during their early stages.

Furthermore, software startups are increasingly obsessed with delivering a software product in an extremely short time besides validating the solution directly with the end user. The use of lean software development methodology and the business models have become popular in software startups, especially in the design of the minimum viable product. Thereby, some problems can be identified by adopting (or not) software engineering practices during the early stages of the startup life cycle.

So the answer we are looking for is related to the further research question: *How are software engineering practices being applied in the software product development of software startups?*

The objective of this article is to present an empirical study on the development of MVPs in software startups located in innovation environments in southern Brazil and providing the answers to the research question. This article is organized as follows: in Section 2, the central concepts that guide this work are presented. In Section 3, the method used is described, and in Section 4 we present the results and the insights of this empirical research. Finally, in Section 5, we state our conclusions.

II. Background

A. Software Startups

Software startups are newly created companies that do not yet have a history of the technological world and operate in a

highly volatile environment with innovative products[4]. They are also called digital startups and find challenges in their insertion in the market. For Steve Blank [1], a software startup or simply startup is a temporary organization looking for a repeatable and scalable business model, while Ries [10] defines it as an institution designed to create new products and services under extreme conditions uncertainty. Startups should have a strategy to develop their prototype and then the product, to then secure a market base for their new products and services, but they have limited access to resources[4]. A startup cannot be confused with a small company. While an established (even small) company has a validated business model, startups are in the pursuit of a business model. Startups have some recurring features [9]:

- Little experience;
- Limitation of resources;
- Various influences;
- Dynamic technology and markets.

The development of a software startup has a different life-cycle than a traditional software company, that is, enterprises that use traditional software development methodologies [19]. Steve Blank [1] defined a four-step process for the initial development of a startup: Customer Discovery, Customer Validation, Demand Generation, and Enterprise Structuring. According to Bosch [2], software startups go through several stages before they reach maturity levels, that is, they have to go through several steps until they can create a business model that works and is feasible.

Usually, decisions related to the development of the product of a startup, such as the use of software engineering practices or not, are part of the executive team of the startup [11]. However, the initial gain achieved regarding flexibility and speed is counterbalanced by the need to restructure the product where the business begins to grow [7].

B. Software Development

All aspects of software production, from early stages to system maintenance, involve specifying, developing, managing and evolving software systems [11]. This procedure is the work of Software Engineering (SE), which arose to solve problems of software systems, aiming to support the development of software using processes, methods, techniques and tools [11]. Techniques such as elaboration and implementation of a computer system consist of the software development process, one of the SE subareas. This subarea aims to transform users needs into a software product capable of solving the user's problems.

The prescriptive models (cascade, prototyping, incremental, among others) are models that prescribe how a new software system should be developed. Prescriptive models are used as guidelines or structures to organize and structure how software development activities should be performed and in what order. Regardless of the process, the following activities are critical to software engineering [11]:

- Software Specification: Definition of functionalities and constraints;
- Design and implementation of software: Production of software complying with requirements;
- Software Validation: Checks if the software meets what the customer wants;
- Evolution of software: The software must evolve to meet the changes requested by the client.

III. Method

The methodology used in this research was based on a Eisenhardt study [5]. In summary, the method steps include a definition of the research, selection of cases, formulation of research instruments and protocols, data collection, analysis by comparison within and between cases, formulation of hypotheses, comparison with literature and theoretical saturation when possible.

A. Getting Started

Initially, we made the definition of Research Questions, which according to Eisenhardt [5] is fundamental to focus on the construction of theory based on the case study. For this case study, we selected startups that have the software base and are located in an innovation environment (business incubator or science and technology park).

Also, we defined factors related to this sample, based on aspects such as:

- Level of maturity: in this case we consider companies that had already developed their product, regardless of whether this product was accepted by the market (market fit) or not;
- Technical partner: important in a software startup is the existence of a partner that understands the technical part and leads the development of the product;
- Interactions with the ecosystem: we consider it important for the object of this research that the number of interactions with the ecosystem of innovation was greater than 1.

B. Selecting Participants

In undertaking case studies, there is the selection of cases. The selection of an appropriate population controls extraneous variation and helps in the definition of limits in generalizing the findings. In case studies related to population, one must be strategic to offer clarified domains of the findings [5].

Using the factors presented previously, our search was for companies that were within the context of innovation, that is, that were inhabiting environments or business incubator or science and technology park. This decision was made because we understood that the analysis of the research question itself could be enriched if we could analyze startups that received the same development opportunities for their development. And in this case, companies in these environments show these patterns. Thus, we chose companies that have their operational bases either in the RAIAR Incubator or the TECNOPUC of PUCRS, Brazil.

To know, RAIAR is an incubator of companies that operates in the lean line of development of software startups and is located in the scientific and technological park, TECNOPUC.

C. Collecting Data

We used more than one source of data and method of collection to increase consistency and reliability: interviews and field observations.

Semi-structured interviews were conducted with 8 companies selected according to what has been previously specified. The interviews script was created with open questions to understand better the relationship of these startups with the ecosystem, as well as the story behind the product they offer to the market today such as the construction of the MVP, the decisions made, or development problems and how were solved which. These interviews were directed to the founders of the digital startups, and this choice was made so that we had the data necessary to understand how the business started and what technical decisions were made to change or not the development of the software product.

Also, an observation made during the incubation process performed by the team responsible for the companies was considered in this research. These observations were based on the structure of the technical team, difficulties faced and reported during the incubation process, such as problems with requirements, verification, system validation, and software architecture.

D. Analyzing Data

A striking feature of any research to build theory from case studies is data analysis, that can be achieved through notes [5]. In this research, we searched for patterns in the answers and categorize them according to the relevance of the proposed theme - software engineering. In this way, we can get an overview of how software engineering is being perceived and used by software startups that are in a technology park.

IV. Results

In this section, we present the achieved results from the field study with interviews, as well as some findings related to the use of software engineering during the development of MVP of software startups.

Among the companies studied the average interaction with the ecosystem was 2.5 years and during this time the majority (62.5%) did not have MVP, that is, software startups began their activities in the ecosystem with the first objective: to develop the minimum viable product. Another important point is the fact that the majority (87.5%) had the technical team (programmers) which shows the control in the software development process.

We could see that among all the activities related to software engineering, some were more evident in the problems presented by the startups during and after the MVP development. According to the interview responses and the areas of knowledge defined by SWEBOK [3], software requirements, software structure and architecture, and software testing are the critical points that will be discussed from now on.

A. Software Requirement

During software development, it is common to have changes related to user requirements. Typically changes occur due to the evolution of the business to which the software is linked and the cost to change a requirement upon software developed is considered too high. However, analyzing the life cycle of a startup (Section II.A) often the customer is unknown to the startup and the requirement management is compromised by this "lack of certainty."

Throughout the research, it was found that a good part of the interviewees (62.5%) use a pseudo agile method to manage the requirements of the MVP. That is, a visual management tool (kanban) and some Scrum practices are used. Basically, *post-its* are stuck on the boards indicating what should be done (new requirements), what the team is working on and what has already been developed. Furthermore, prioritization is a common practice among startups.

Asking the startups more about requirements management, 100% indicated that many requirements are not managed and/or documented, being passed on verbally either by the key user or the startup leader. When questioned about the reason for using some practices and not others, all of them indicated that the time they have should be used in product coding and avoiding the development process to be bureaucratic.

Our first finding comes up from the situation described before.

Finding 1 *Adoption of requirements management techniques lead to a significant increase in development time.*

A second finding can be defined as follows:

Finding 2 - *Non-documentation and/or requirement management does not interfere with the quality of the MVP developed.*

B. Software Structure and Architecture

Since most of the entrepreneurs in each company have a technical background, at several moments in the interviews, it was stated that there was the need for an improvement of the software structure and architecture. However, much of the development was carried out without adequate planning and without the use of structures and coding standards that allow the growth of the system (either in the number of functionalities or the number of users).

Table I presents the problems with software structure and architecture faced by the surveyed startups.

The following finding related to the software structure and architecture can be formulated:

Finding 3 - *A poor definition or non-definition of a software structure and architecture in the initial phase of a software startup exponentially increases the company's technical debt.*

C. Software Testing

Despite these numbers, the technical team in its entirety did not use software testing techniques in the construction of the first version of the system, leaving to the end user this responsibility. In the following versions of the system, this scenario changed considerably, 75% used some software testing technique.

TABLE I. PROBLEMS WITH SOFTWARE STRUCTURE AND ARCHITECTURE

Problem	Description
Coding language & associated IDE	The choice of programming language may affect future extensions of the software and integrations with existing solutions in the market.
Database	Need to choose a database that has an active user community, allowing a search for solutions to problems encountered while using MVP.
Web host	Web hosting is critical for the solutions developed by software startups. It needs to be an affordable and secure framework.
Code Deployment tool	Always making software changes and the associated implementations that come with it need to be controlled to increase team productivity and quality.
Security approach	Usually indicated as an important item by the startup team but ignored during development (using the security solutions offered by the web host).

In Fig. 1, we can see all different ways that these startups had been performing software testing in their MVP life cycle.

- Using a pilot client: It is considered here the system test performed by a group of end users, prior to deployment, to provide feedback to the product development team.
- Unit tests: a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.
- Functional ad-hoc tests: software testing performed without planning and documentation.
- Specialist tester: software test performed by a business specialist associated with the startup.

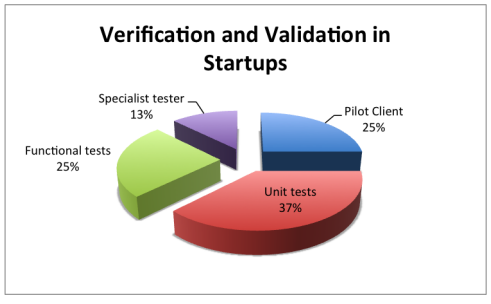


Fig. 1. Software testing in Startups

The way that the startups are performing software testing activities during their MVP life cicle leads to a fourth finding:

Finding 4 - *Structuring or formalizing a software testing step will increase the market acceptance of the requirements developed.*

V. CONCLUSION

In this paper we presented the results from an empirical study on the development of MVPs in software startups located in innovation environments in southern Brazil, mainly analyzing the problems encountered during the software development phases.

Based on the sample of startups studied, we verified that requirements, software structure and architecture, and software testing are critical and should be more detailed. The four findings defined in this paper should be better studied in order to find insights that can improve the development process used by startups.

The results showed in this paper provides an initial step for understanding what happens in the early phases of any software startup, mainly related to software engineering activities. A few limitations must be taken into consideration in this study. First, the sample size was small. Second, interviews were conducted only with startup founders. As future work we intend to consider a bigger sample of startups as well as a more diverse set of interviewees.

ACKNOWLEDGMENT

This work was partially supported by grants from BEPiD/PUCRS (Brazilian Education Program for iOS Development).

REFERENCES

[1] Steve Blank. *The startup owner's manual: The step-by-step guide for building a great company*. BookBaby, 2012.

[2] Jan Bosch. Speed, data, and ecosystems: the future of software engineering. *IEEE Software*, 33(1):82–88, 2016.

[3] Pierre Bourque, Richard E Fairley, et al. *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press, 2014.

[4] Henry Edison, Xiaofeng Wang, and Pekka Abrahamsson. Lean startup: why large software companies should care. In *Scientific Workshop Proceedings of the XP2015*, page 2. ACM, 2015.

[5] Kathleen M Eisenhardt. Building theories from case study research. *Academy of management review*, 14(4):532–550, 1989.

[6] Henry Etzkowitz. The evolution of the entrepreneurial university. *International Journal of Technology and Globalisation*, 1(1):64–77, 2004.

[7] Carmine Giardino, Xiaofeng Wang, and Pekka Abrahamsson. Why early-stage software startups fail: a behavioral framework. In *International Conference of Software Business*, pages 27–41. Springer, 2014.

[8] Per Lenberg, Emil Alégroth, Robert Feldt, and Lars Göran Wallgren Tengberg. An initial analysis of differences in software engineers' attitudes towards organizational change. In *Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 1–7. ACM, 2016.

[9] Nicolò Paternoster, Carmine Giardino, Michael Unterkalmsteiner, Tony Gorschek, and Pekka Abrahamsson. Software development in startup companies: A systematic mapping study. *Information and Software Technology*, 56(10):1200–1218, 2014.

[10] Eric Ries. *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Crown Business, 2011.

[11] Ian Sommerville. *Software Engineering*. Addison-Wesley Publishing Company, USA, 9th edition, 2010.

POQAS-S: a Novel Programmer-Oriented Online Question Answering System With Semantic Comprehension

Feng Guo, Guangquan Xu*, Ning Zhang, Kaili Qiu
Tianjin Key Laboratory of Advanced Networking (TANK), School of Computer Science and Technology
Tianjin University
Tianjin, China, 300350
*Corresponding author: Guangquan Xu, email: losin@tju.edu.cn

***Abstract*—To improve the ability and efficiency of acquiring knowledge for programmers, programmer-oriented online question answering system (POQAS) has been advanced recently. However, current POQAS systems are short of semantic sensing ability. To solve this problem, this paper proposed a novel programmer-oriented online question answering system with semantic comprehension (POQAS-S), which is based on the TuringOS. Since semantic comprehension is introduced, our POQAS-S system can understand the input information more precisely. Further, POQAS-S is able to give a more precise answer to users than traditional POQAS systems. In this paper, we developed a concrete POQAS-S system on the android platform, which demonstrated well in contrast to other traditional POQAS systems in a more precise answering providing. Our POQAS-S can be referred to develop more other similar POQAS-S systems besides out built android APP in this paper.**

Keywords —Semantics, Knowledge base, Android, and Chat robots.

I. INTRODUCTION

Android is a free and open source operating system which is based on Linux, mainly used in mobile devices, such as smart-phones and tablet computers, characterized by Google and Open Handset Alliance. The advantages of Android are: open, rich-hardware, and easy to develop, also the mobile application of Google is a divine place for developers in the heart [1].

Information interaction would never be stopped since the beginning of humanity. Accompanied by the development of social science and technology, it has constantly changed people's form, and always occupies an important position in people's life. As a kind of extension about global information, Mobile terminal network service has greatly changed our life. In today's mobile tide, the mobile terminal as carrier of various service applications is brought to shaking changes our lives [2].

Although there are a large number of information service applications and successful cases on Android platform, due to various reasons, many problems are still existing: the function of Baidu applications is powerful but too popularization and not meticulous enough; Twitter has a broad field of information but did not realize information packaging and there are lots of redundant information; the information service of WeChat of the public account is not convenient to find, etc. So, a new, distinctive, concise and powerful for specific groups of APP is in urgent needs.

DOI reference number: 10.18293/SEKE2017-190

The purpose of this task is to make up the current information service applications' problems and to absorb their advantages, and developing our application's own characteristics: this application is based on a good interface and with the function that increasing the information storage; In order to satisfy the programmers' demand, adding the push function at the foundation of the reading program articles, using SMS verification functions to make users log in, which makes the APP easy to use. The "intelligent response to question messages" is the core characteristic.

This design provides an APP for programmers which are compatible with current mainstream features, simple and powerful, focused on providing users with replying related information with high efficiency, reducing the redundancy of searching information, and improving efficiency greatly [3].

With years of development, Android has become a platform, and an ecological system. Now, Android has been widely used in mobile field. In the past, Android versions updated too fast and the compatibility issues make the Android Market share not big, but with its stability improvement and mobile intelligent industry's vigorous development, more and more developers began to enter into the Android world, both in hardware and the software system. All the signs show that, Android is worthy of the name "king of mobile".

Because of the portable mobile devices, application of information service has been rapidly developed. An information service application can help all kinds of people know more about the world and promote people to get the information what they want accurately [4].

In this "data is wealth" era, the success of information servicing applications can bring programmers great opportunities and attract domestic and foreign Pinnacle Technology Corp, free developers, the elite from all walks of

life. But in the face of huge wealth, competition is more intense compared with other areas.

The main content of this paper is the basic research of a user to an information service application and realize the characteristic functions of the application [5]:

- User management: achieve user registration and log in, retrieve the password,
- messages log in: send the verification code to the phone number, in the time limit of 60s and in accordance with the corresponding verification code to log in the system, the web server can be listening to the corresponding log in information, the successful logging in is recorded, then when you log in next time, you do not need to enter the verification code.
- Information query: it is the core function, according to the transmitted message, it can find the corresponding answers, which is related to NLP knowledge base. for example, the question encountered in various programs can find answers by this function, also the server can set the information ambiguity [6].
- Article recommended: the server update an article which is related to IT every day, and pushed to the APP's interface, then, the users can view.
- Information storage: the corresponding text, pictures, or video can be stored, viewed and deleted, stored in the SQLite database, and it can use the user's gesture password information to protect.
- Information browsing: the server in the APP in accordance with the classification of the IT technology provides a variety of knowledge encyclopedia information database, users can view the corresponding information; the information database server can be updated in real time.
- Push function: this APP can push the users' mobile phone information in IT aspects.

II. RELATED WORK

A. Android technology

In the domestic mobile phone market, android operating system occupies the most market share. Android based on Java programming language, makes the interfaces and functions take with an endless stream of changes [7].

1) The advantages of Android development:

a) great openness: Android is opened with source, the system source code can be used by the public, and its modification, distribution is not restricted by the permit.

b) high degree of freedom: the system has a variety of simple and practical widgets, developers can follow their habit to develop application, showing their own ideas and thoughts.

c) advantages of development language: Android use the Java language, because of it, Android development is simple, efficient, rich in resources, and can use lots of frameworks, also with the advantages of connecting with the WEB server seamless,suitable for current application software industry with increasing competition.

B. Turing engine

TuringOS is an intelligent operating system which can simulate human feelings and thinking mode. It has the interaction ability which is mostly close to the human, including emotional calculation, thinking and self-learning three engines.

Turing robots have skills to learn so as to suit for all kinds of knowledge fields. Turing engine is based on the NLP knowledge base, leading to artificial intelligence technology and user private knowledge base's effective union. In order to meet the life, and business needs, robot owners can import knowledge bases, and injecting "exclusive content" for their own robot [8].

This APP is based on the engine, with the realization of the function of the core of the question and answer, the difficulty in the programming is the engine's access and design of the APP simulating the box that can accept the online message contents.

III. REQUIREMENT ANAYSIS

A. System performance requirements

Client response: ensure the client's priority with response of user's operations, so as to provide a better user experience; user's log in, interface conversion and other response time control within 2S, message receiving and sending response controlling within 1s merely [9].

The server response: with the queue model of request to achieve the effect of response prioritized and treatment delayed, cluster services and provide more stable service; the operation to the database should is control in less than 5S, manner to the client returns failure information if the operation failed in timely, and rollback data; solve server problem by itself as far as possible, send returning message to the client.

B. System reliability and availability requirements

Data transmission reliability: To ensure the transmission speed, the conventional data use J-son string transmission; data response reliability: control the way to the database by use of database connection pool technology, reading and writing the database to ensure the efficiency of the database response; client interface usability [10]: clean and tidy interface design, does not appear to provocative color and picture, no embedded advertising, improved functional tips and easy to use;

Menu design usability: reduce the multi-layer nested menu, menu identification is easy to be understood, prefect menu, eliminate the difference of function and instruction; interaction design usability: for a variety of operating, providing a more obvious response, such as text, the box, the progress bar, etc. To avoid stiff interactive or non-interactive, as the first priority, user's response provides a good sense; interface update requirements: the design of the interface configuration by use of form of configuration , easy to modify and follow up; function update requirements: the function of modular design, to ensure that the system's high scalability, easy to follow up function; server needs to expand the server: when the client sends a request to the server, by forward the request to intercept

and forward the request to adapt to the subsequent expansion; database design requirements: the table of the database and the field obey unified standard design, in order to follow the increasing amount of data, it may be used to separate the implementation of reading and writing [11].

C. Feasibility analysis

Analysis of the feasibility of the task is the necessity and feasibility of the application of project development. Necessity comes from the urgency of the task, but feasibility depends on the realization of the application system of the resources and conditions. The project need set up on the basis of preliminary investigation.

D. Technical feasibility

Android is one of the most popular mobile terminal platform, it has good usability and ease of development, resource rich and so on. At the same time, many incomparable advantages for developers to provide strong function and the abstraction, so that developers can avoid repeated low-level work and focus on business and achieve.

Java, as one of the most widely used business language, has the characteristics of high efficiency, stability and rich open source framework. It plays an irreplaceable role in server programming.

This project uses the Android client, and servers which are based on the Turing robot engine. Being use of interface framework it can be an architecture level of good design of the system [12]:

- Client as the view display layer, based on the use of Eclipse Java prepared. On the one hand, Android platform for the development is simple and efficient. on the other hand, rich open source information of Android platform can provide us with a powerful boost;
- As control layer, Server-side logic part based on eclipse using Java, through the use of Turing robot engine server can make our development focus on business. on the other hand, Java, with the stable and efficient merit and rich and open framework, with the popularity of spring and Struts2 framework which greatly facilitated the development to us;
- Above all, the client uses the Android platform application development to realize high efficiency, on the other hand, with the help of Turing engine and open source framework, the server can gain a robust architecture and efficient development, in terms of technology, it is feasible [13].

E. Economic feasibility

The cost of this project included two parts: the cost of the hardware device such as a server and the mobile terminal hardware such as a test machine.

At present, a personal PC, personal mobile terminal intelligent equipment is enough to meet the requirements. As college students, these two devices can be very convenient to get in, even if the repurchase, the burden of cost is not great. And application software system, due to the characteristics of Java and Android, a large number of excellent open source frameworks and the servers can allow us to use it free without charge, so the cost are allowed to ignore in this hand. After the completion of the project, we can through upload app to store and get download traffic revenue consequently [14].

F. Risk factor control feasibility

As a project, this project can access to information from the knowledge base and Internet when meeting some mistakes, which to ensure that the project is without errors.

IV. POQAS-S SYSTEM BASED ON ANDROID

A. Overall design

Overall design is mainly to transform the logic model of demand analysis into the physical model of the overall design, and set out to achieve the demand of the system. Overall design is not only to design the system of the overall function structure, but also the relationship between these modules and signals between them. Because modules are independent of each other, so we can separate the processing such as code design, write, test and modify, thus reducing the possibility of error spreading in the module, improving the reliability and maintainability of the system. And then for the design of database, which is an important part of the system, a good database can make the system run more smoothly. At the beginning of the development of this system, we should design each data set in the database, and the standard relationship between them.

B. System function design

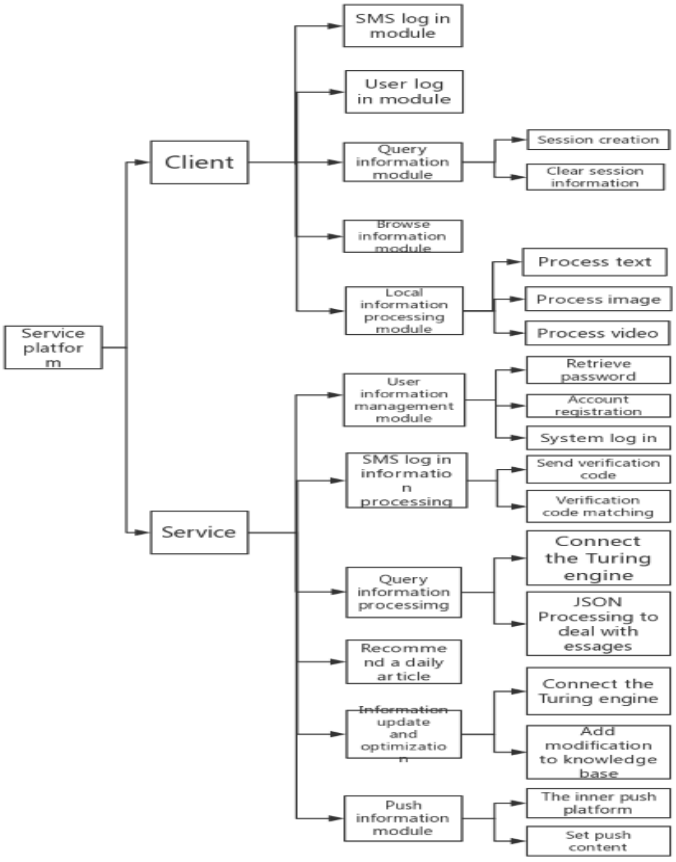


Figure 1. The system function module chart

C. Android client

1) Query information module

The module is mainly responsible for sending request information. Getting request information from the client activity, changing it into corresponding *Json* string format, and then these strings assembly to the corresponding URL parameter way. The HTTP can get access to this URL and send the information to the server.

2) Browse information module design

The module is mainly responsible for the server to receive and parse information.

On the server side, the results transported by *j-son* strings (transformed by *g-son* object) are complied with the HTTP agreement. The client gets the corresponding *j-son* string from input-stream complied with the HTTP agreement, definitely the *j-son* string also transformed to the corresponding java class object. The parsing process needs external jar package (*g-son* package). The corresponding Java objects are used for displaying users' information on the client side.

3) Local information storing module

The module is mainly responsible for managing the information stored on the client side, such as adding, deleting, and modifying, inquiry and etc. information including text, pictures, videos can be stored for users. Users can call the camera and videos hardware interface on the client side to save the information they wanted, and to remark the information which is stored in the *Sqlite* database, and by generating the related thumbnail in the style of list view. So, users can check and delete information stored at any time.

In order to ensure the security of stored information, Gesture passwords lock is used in this module. Users can set the gestures passwords before they used. At the next time, when users use the module, they must be verified by gestures passwords. Of course, when you are verified successfully, it can also be closed. Gesture passwords have set the boot menu, making users setting conveniently, and through this interface by setting up the boot sequence of gesture passwords graphics, user-friendly and beautifully.

4) Information processing module

a) Users logging in module

The client is responsible for the judgment of inputting information which is legal or not, the system is responsible for the information verifies, especially for determining whether the inputting information matched completely the information in the database. If the information is correct, it will be transformed to a user class object, which is through the socket transmissions, sending logging request to the server. Server will parse the user object of user information, and then check the information of the user existing or not, if the user information exists, then the server returning the message that is logging in successful to the client. The client jumps to the next Activity. If the user information d does not exist, then the server returning to the client: the user name or password error. The client prompts users to input the user information again.

b) SMS landing module

Users can use the phone number, and then enter the verification code, its limit time is 60s, while, if the information is matched, this action is successful; otherwise the system pop the message box to ask users whether or not to send verification code again. If there are records of a successful logging in, the mobile phone number will be preserved, and users can skip the stage of verification code next time.

The function named *setRegisterCallback* is responsible for handling the response of the related event. Firstly defining a *RegisterPage* class which is imported by an external SDK package, but method named *afterEvent* must be rewritten: if the country and the phone numbers are inputted, they will be uploaded to the SMS by using the method of hash map server, and *submitUserInfo* upload function also needs to be rewritten: The function named s*ubmitUserInfo* needs to be added two additional parameters named registering nickname and random UID (using mathematical random function).If the verification is successful, then jumping to the main interface of the app, and saving the information successfully. If the validation fails, jumping back to the logging interface again [15].

c) Update information and optimize the module

After logging in the system, the designer can update the knowledge base which is of knowledge module, and the existing knowledge base module including data structure, algorithm, programming language, computer basis, application software system development, mathematical and logical ability module can be altered. The changing of corresponding modules is side in Turing engine, which is the server side. For instance, modifying the server NLP knowledge base can realize the information update.

d) Push information module

The module is mainly responsible for information recommendations. Developers will put the important programming information as a hot topic to users who use this app. The specific push messages are defined in the platform of JPUSH garage, through an external SDK to use the platform to implement the corresponding information push operation, this process plays an important role for users to increase the programming knowledge which is the core for the aim of this system.

e) Recommended daily article module

The server will update information of corresponding NLP knowledge base every day to achieve the requirement of a good programming article which is recommended daily, the corresponding good articles were collected, and added in the NLP knowledge base. Users will transmit the request information encapsulated good daily article to the server, and then, the server obtains the information, and queries the corresponding of NLP knowledge base. Note: due to the specified format returned and the length of paper generally are too long, so we should be created the additional buffer pool space to realize the information buffer [16]. There are two ways to realize the article pushed, one is using the existing JPUSH platform to display on the client, the other is using Turing engine to push.

f) Socket transmission

Using the object of class of the Socket to write data in the input process. Firstly, server-side declares a *Server-Socket* object and specifies the port number, and then call the *accept* method of *Server-Socket* to receive the client's data. When there is no received data, the method *accept* is in blocked state. Once receiving the data, the method *read* deal with the received data through the input-stream. The client creates a Socket object, which specifies the server IP address and port number to read the data from the input-stream, by this, the client obtained the data from server. Finally, Client sends the data which should be sent to the output-stream. Namely it is time to precede socket data transmission of TCP protocol. Server-side logic process: when the information of users who log in is empty, the user object which the server obtained will return the prompt message of empty logging information. Otherwise, according to the logging information submitted by user object, Server-side invoke *log* method of server-side method *User-Dao* to log in.

g) Registering

In this module, when the user firstly enters the logging page, he must fill in the blanks which are "user name", "phone number", "password" and "confirm password", "your spouse or father or mother or children or company boss or your primary school teacher's name". The client access to these information, firstly checkout the password in the third blank is same with the second blank. if it is not consistent, then the client reminding the user to fill in it again. The server would get the information until the client conforms the information to meet the requirement of the standard.

h) J-son parsing module

The module is mainly responsible for the *J-son* object which is to deal with a request transformed from Client parsed into Java objects, Server obtains *J-son* object from the client request information, according to the current request interface set to find the corresponding tools in this module, which parsing the *J-son* object to a given Java object, so as to give back the other requests interface to use.

V. CONCLUSION

This paper introduces an APP created by us. The system was basically completed, but due to the writer is still insufficient in terms of technology and design ability, some function implementation could be optimized. Firstly, the pushing information module could be changed by using the latest frame. Secondly, the knowledge base could use k nearest neighbor algorithm in machine learning to improve the match rate in searching. Most importantly, more measures should be used to check the safety of our system because we are more interested in speed than safety and reliability.

According to our experiments on questions in programming area to be asked to our system, it achieves 85% right solved rate (comparable to other POQAS systems) and 1.5X faster than these systems, which makes it the great solution viable for use cases of POQAS systems. But we still need to update knowledge base constantly to improve the right solved rate,

VI. ACKNOWLEDGEMENT

This work has partially been sponsored by the National Science Foundation of China (No. 61572355) and Tianjin Research Program of Application Foundation and Advanced Technology under grant No. 15JCYBJC15700, Xinjiang Corps Technology Market Association project under grant No. 2016GKF-0693.

REFERENCES

[1] Zong Woo Geem, Joong Hoon Kim, G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," in SIMULATION, vol. 76(2), pp. 60–68, February 2001.

[2] Moles CG1, Mendes P, Banga JR, "Parameter estimation in biochemical pathways: a comparison of global optimization methods," in Genome Research, vol. 13(11), pp. 2467–74, 2003.

[3] Alexander Budanitsky, Graeme Hirst, "Evaluating WordNet-based Measures of Lexical Semantic Relatedness," in Computational Linguistics, vol. 32(1), pp. 13–47, 2006.

[4] Alan Aronson, "Effective Mapping of Biomedical Text to the UMLS Metathesaurus: The MetaMap Program," in Proceedings of the AMIA Symposium, pp. 17–21, 2001.

[5] YM Wang, TMS Elhag, "Fuzzy TOPSIS method based on alpha level sets with an application to bridge risk assessment," in Expert Systems with Applications, vol. 31(2), pp. 309–319, 2006.

[6] Huan Liu, Lei Yu, "Toward integrating feature selection algorithms for classification and clustering," in IEEE Transactions on Knowledge and Data Engineering, vol. 17(4), pp. 491–502, April 2005.

[7] A Goldsmith, SA Jafar, I Maric, S Srinivasa, "Breaking Spectrum Gridlock with Cognitive Radios: An Information Theoretic Perspective," in Proceedings of the IEEE, vol. 97, no. 5, pp. 894–914, May 2009.

[8] Clément Sanchez, G. J. de A. A. Soler-Illia, François Ribot, T. Lalot, Cédric R Mayer, V. Cabuil, "Designed Hybrid Organic−Inorganic Nanocomposites from Functional Nanobuilding Blocks," in Chemistry of Materials, vol. 13(10), pp. 3061–3083, 2001.

[9] Igor Luzinov, Sergiy Minko, Vladimir V Tsukruk, "Adaptive and responsive surfaces through controlled reorganization of interfacial polymer layers," in Progress in Polymer Science, vol. 29(7), pp. 635–698, 2004.

[10] Uttam C. Paul, Despina Fragouli, Ilker S. Bayer, Athanassia Athanassiou, "Functionalized Cellulose Networks for Efficient Oil Removal from Oil–Water Emulsions," in Polymers, vol. 8(2), pp. 52, February 2016.

[11] Ling Wang, Quan Li, "Stimuli-Directing Self-Organized 3D Liquid-Crystalline Nanostructures: From Materials Design to Photonic Applications," in Advanced Functional Materials, vol. 26(1), pp. 10–28, November 2015.

[12] Zan Guangtao, Wu Qingsheng, "Biomimetic and Bioinspired Synthesis of Nanomaterials/Nanostructures," in Advanced Materials, vol. 28(11), pp. 2099–2147, 2016.

[13] Kumar Dipesh, Chatterjee Kalyan, "A review of conventional and advanced MPPT algorithms for wind energy systems," in Renewable and Sustainable Energy Reviews, vol. 55, pp. 957–970, March 2016.

[14] Holtgraves Thomas, Han Tai-Lin, "A procedure for studying online conversational processing using a chat bot," in Behavior Research Methods, vol. 39(1), pp. 156–163, February 2007.

[15] Imran Ali Shariq, Kowalski Stewart James, "HIP - A Technology-Rich and Interactive Multimedia Pedagogical Platform," in Lecture Notes in Computer Science, vol. 8523, pp. 151–160, 2014.

[16] Kang Ah Reum, Kim Huy Kang, Woo, Jiyoung, "Chatting pattern based game BOT detection: Do they talk like us?" in KSII Transactions on Internet and Information Systems, vol. 6(11), pp. 2866–2879, 2012.

Logical Tree and Complex Event Processing in Power Systems

Juntao Li
Peking University
Beijing, China
lijuntao@pku.edu.cn

Yibo Yang
Peking University
Beijing,China
iboing@163.com

Yu Huang
Peking University
Beijing,China
hy@pku.edu.cn

Yiming Lu
Department of power distribution network China Electric Power Research Institute Beijing, China
luyiming@epri.sgcc.com.cn

***Abstract*—In virtue of the sharp increasing scale of distribution network and remarkable improvement of network complexity of power system, it is more challenging to monitor the actual running situation of the power network and to diagnose fault. To add insult to injury, man-made operation mistakes and inadequate situational awareness push the power system into a tight corner. This research presents a new approach, which could achieve efficient and sufficient situation awareness. Firstly, tree form structure was adopted in this model. Then the complex events processing engine is introduced. As a result, a comprehensive model is proposed in this paper, through taking data-based knowledge and model-based information into account. In the end, the software framework and results of simulation experiment is provided**

Keywords—situation awareness; complex events processing; trees; power systems; data-based model

I. INTRODUCTION

Mounting number of network nodes at present have threatened the overall stability of power system, posing a conundrum for researchers to monitor various faults. There are already numerous incidents all over the world that are relative to power systems, one of which is the large-scale outages happened on august fourteen, 2003 in North America, as a result of which tens of millions have been affected and even paralyzed the urban public transit [1]. Another case of blackout occurred in September 2003 in Italy which had grabbed the news headlines for a period of time [2]. The most important element in these cases is devoid of real-time diagnoses and precise situational awareness in system of power networks. Accordingly, methodologies that are intimately bound up with situation awareness. Apart from situation awareness models, decision-making support methods offer effective tools for researchers and operators. With regard to decision-making support methods, decision tree algorithm is the most commonly used in practice, consisting of two main categories of trees: regression trees and classification trees [3]. The application of such a tree structure makes it possible to accomplish highly efficient and accurate anomaly detection and further rapid classification as well as ultimate predictions.

In view of this, a new model of software architecture compatible with the existing power systems was proposed in this research. The main contents of this paper are as follows: the first part introduced the background information of situation awareness and its application in power system; the second part of the paper discussed related works, including implementation of regression trees and decision tress in power system, model-based approaches of situation awareness , data-driven algorithms, generic frame-structure and practical tools; in the third section, a reasonable and complete definition of situation awareness has been given; in part four, the process of situation in power system was been illustrated; in part five, specific algorithm of situation awareness was proposed; the six part of this paper indicate the results of simulation experiment; the seven part of this study is the conclusion that sums up this work and mentions a few problems to be resolved.

II. RELATED WORK

The theories and problems as well as application of SA in power networks have been researched, consisting of the sources of inadequate SA, an information system that is applicable to complex system, and a general architecture of software. In study [4], the standards and tools of SA in power networks have also been given, as a result of which perception levels of system operators were enhanced obviously. From another perspective, a novel approach, a typically data-driven theory, has been introduced to cognize the status of power networks, which is mainly based on the random matrix theory [5]. Another key technology achieved in electrical networks is decision-making tree due to the fact that it possesses a unique characteristic compared with other machine learning algorithms [6, 7, 8, 9]. Meanwhile, the performance estimation of CEP in real-time network monitoring turned out to be very attractive, since it has simultaneously real-time access to data, fast speed query, and effective data reduction [10]. In this research, hence, a novel method that is a combination of CEP technologies and effective trees inference structure showed excellent capability of situation awareness in power systems. Meanwhile, through taking into account information obtained by data-driven approach, a variety of hierarchies of control information is quickly pushed to corresponding users in power systems.

DOI reference number: 10.18293/SEKE2017-189
This work was supported by the Major State Research Development Program of China(2016QY04W0804)

Besides, another definition of situation awareness in power gird was given in this paper and also the software based on frame-structure illustrated later was accomplished.

III. DEFINATION OF SITUATION AWARENESS

Situational awareness reasoning model for distribution network was built by using the combination of Analytic hierarchy process (AHP) and B-tree-like structure, and users can also evolve this model. Through taking different rules sets, the reasoning process of model can be achieved from down to up and vice versa, to implement disaster situational awareness and trace corresponding resources in the converse process. Moreover, when it comes to the process of modeling, each element in each of the AHP layers is defined as a single node. By connecting all the nodes, the B-tree for situational awareness was built. Fig. 1 showed the object of model elements.

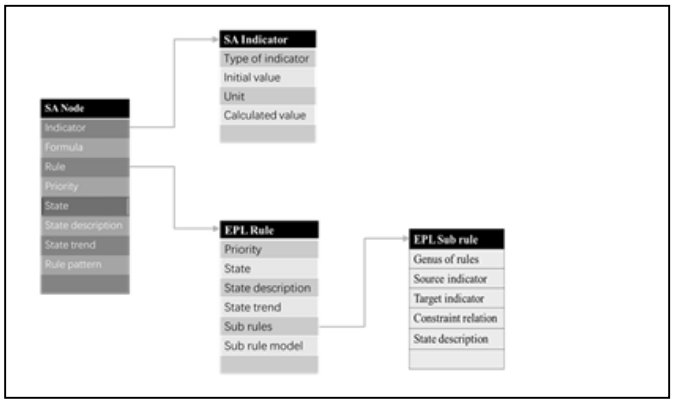


Figure 1. The object of model elements includes nodes, indicators, rules and sub-rules

In addition, relative data normalization of situational awareness is as follows. First, basic elements and constraints of the construction of the frame are defined, which each basic element represent element model (node). Eventually, tree of perception model was constructed through using rules to connect individual nodes. Fig. 1 showed structural design of each node, and structure of rules and indicators.

Constraints of nodes are as follows: each node contains one or more rules; each rule comprises one or more sub-rules; there exists only one relation of 'and', 'or' between rules in the same layer, so do sub-rules; state values of rules and sub-rules consist of two types: true and false; through applying 'and' operation and 'or' operation in the same layer, the state value of sub-rules and rules could be aggregated into the state values of upper layer; reasoning and judgment of sub-rules is based on the statues values by comparing values of source indicators and target indicators; the function of and-or-invert could be reasonably decomposed into combination of 'and' and 'or' operation.

The essence of situation awareness model is ontology model and knowledge, which makes it possible that users could create the classification of situational awareness model with the model library. Users can manage their own ontology model and knowledge.

Situation awareness model contain the following attributes: meta-model knowledge, ontology knowledge of rules, ontology knowledge of indicators, other attributes description. Ontology knowledge of rules can also include knowledge of sub-rules, formula, and indicators. Ontology knowledge of formulas can also contain sub-formulas and indicators. The system further abstracted rules into atomic events and composite events. While formulas are used as the expression of events detection, indicators are defined as monitoring parameters of events detection.

IV. PROCESS OF SITUATION AWARENESS

Dynamic hierarchical model in early warning is based upon analytic hierarchy process and trees structure, which build the relationship between key factors and nodes of the tree. As a result, hierarchical early warning model of power grid security can be established, which can be used for dynamic hierarchical warning of power system risk. Flow diagram of situation awareness is indicated in Fig. 2.

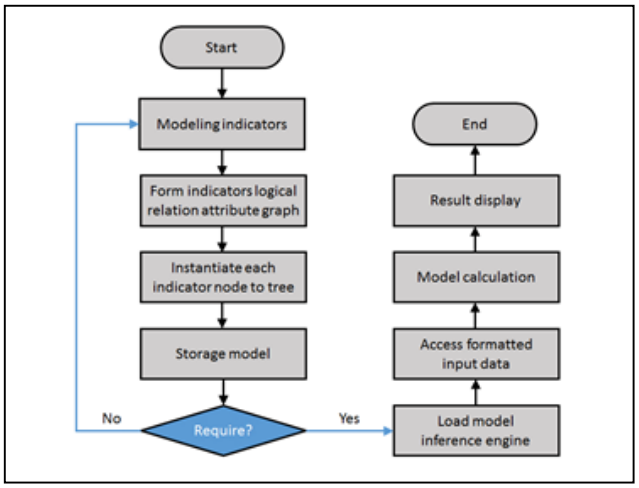


Figure 2. The flow diagram of situation awareness

This model consists of the following steps:

1) Establish the indicators system of power networks security risk early warning: According to the collected data in the monitoring system of power networks, the indicators preliminarily selected such as natural disasters and man-made mistakes.

2) Construct the graph of tree structure with layered nodes and child nodes that is corresponding to stratification of regions: Each leaf node of the tree represents a county. Their parent nodes represent corresponding cities which are larger regions than counties, while the region of a province is expressed as the root node.

3) Set the indicators and formulas as well as hierarchical model of power networks security risk early warning involved in each node for each single layer of the tree: The setup method of hierarchical model includes the following steps: The first stage is to define variables, and variable valuations are obtained from further calculation by using those relative formulas. The hierarchical model can be optimized on account of variable valuations, which the layers of hierarchical model are consistent with regional layers. This model is established and stored. The model that has already been stored also supports modification operations.

*4) Each node of hierarchical model is marked by different colors:*In practical application, the implementation of the operations are as follows: The object of previously created early-warning model is initialized with tree structure, and the object of indictors in the warning model is also loaded in memory; Certain types of data are obtained from the detection system of power gird as input data; Matching nodes of input data are found according to different indicators; input data can be computed use formulas within coincide node; various values are labeled as different colors by means of rule-based judgement of each node; different colors represent different levels of risk; red color indicates that the node is at risk, while green color indicates devoid of risk; when a specific node of a layer turns red, homologous risks are warned.

V. SPECIFIC ALGORITHMS

A. Complex Event Processing Based Situational Awareness

The situation in the short term and ultra-short term can be predicted on account of the result of data understandability. Predicting consequences of relatively low level is primarily focused on deterministic prediction in the case of a single fault, while higher level yield the probabilistic load forecasting results of all possible circumstance. Deterministic prediction presents exact single-point prediction, while probabilistic prediction provides confidence level and probability of predictable variables.

In addition, the situation awareness is implemented in the rolling way which means predicting outcomes in the past would be the present situation as time goes by. Therefore, future predicting consequences should be amended by current results of data cognition and situation awareness.

B. State Machine Processing

Detection and monitoring of composite events are implemented based on the combination of rules, indicators and initial values. Workflow-based data is propagated to real-time indicator data of each threshold node. Current situation is acquired from both model and data. The approach of complex event processing is used to analyze the variation trend and relationship of data for the reason of achieving situational predication. Through taking the current and future state of the model into account, users can set up rules and model to realize intelligent early-warning.

C. Establishment of Inference Rules

Inference rules can be abstracted as events in the system, as a result of which atomic events and composite events represent rules and rule set respectively. The implementation of rules is as follows: ontology knowledge of rules is established; Meta knowledge and essential attributes are maintained; indicators are added to rules; indicators names and initial values as well as description information are maintained, which the indicators are available from indicator base or user-defined indicators; rules bind to sub-rules, which all these rules are selected from existing library. Judgment outcomes and status value (true/false) of rules and sub-rules are the main reference of state changes, which status values are obtained from the 'and/or' judgment of rules based on the node models. According to state values of rules or nodes, the node information of next level could be acquired so that situation reasoning and awareness could be accomplished, as illustrated in Fig. 3.

Situational awareness reasoning process

```
Set of ordered nodes A;
access root node v;visited[v]=1;// before the algorithm is executed visited[n]=0
w=first adjacent node of v;
while (w exists)
        if (w not accessed)
                start from vertex w recursively execute this algorithm;
        w= first adjacent node of v;
                A.add(w); //put the tree nodes in set A form the lowest layer
void reasoning(b,M)   // inference node process
    {
        isreasoning=false; //whether need to reason
        if (b node with children){
                if (b relation between sub nodes 'or'or the relationship is not set){
                        if (b any child node meets the requirements) {
                                isreasoning=true;
                        }
                }else if (b relation between sub nodes 'and') {
                        if (b all sub nodes meet the requirements) {
                                isreasoning=true;
                        }
                }
         }else{// no child nodes
                isreasoning=true;
        }

         if(isreasoning){
                if(b node needs to access the external interface){
                        c= results of external interface;
                 }else{
                        c=results of local reasoning methods;
                 }
         }
         M.put(b,c);//put inference result in set M
    }
Inference result set M;
for (iterate set A){
    b=a[n];
    reasoning(b,M);// put inference result of nodes in set M
    }
void recursionNode(node ){
    if (node has child){
        d=child node;
        recursionNode(d);
```

Figure 3. The specific algorithm of situation awareness

Knowledge can be adversely traced back to its sources on the basis of the reasoning results of the model. And relevant knowledge of rules and indicators as well as meta-knowledge in the process of inference could be adversely traversed and can be presented to the user. Through considering the correspondence between complex events and specific rules, and saved incidence relation between knowledge of rules and ontology knowledge, all of the knowledge can be traced back. The algorithm of adverse tracing is given in Fig. 4.

Algorithm of adverse tracing

```
access root node v; visited[v]=1;//before the algorithm is executed visited[n]=0
inference result set M;// obtained from prior reasoning
void reverseTracing(node ){
        if(m.get(node)) {//the data of the joint
                d=child node;
                print node;//print node information of problem nodes
                recursionNode(d);
        } else {
                print node;// print node information of normal node
        }
}
reverseTracing(v);
```

Figure 4. Algorithm of adverse tracing

VI. SIMULATION EXOERIMENT

Data for reasoning includes weather (temperature, wind velocity), power lines (length, type): Rule of node wire length is that the length of wires is more than l0 kilometers; rules of

node temperature is that the temperature is below three degrees; nodal wind speed is greater than 7m/s; the type of power wires is overhead line; overhead lines icing will occur when the length of overhead lines is greater than 12 kilometers and wind speed is greater than 10m/s.

Data rules: Incoming weather data:

{"data":{"weather":[{"ID":"weatherID1","temp":1,"windspeed":6,"eleLineID":"eleLinedID1"}]},"finish":"false"}

Incoming wires data:

{"data":{"eleLine":[{"ID":"eleLinedID1","len":150,"overhead":1}]},"finish":"true"}

'finish' represents whether the data is introduced completely. If completed, reasoning process is started. The reasoning results are shown in Fig. 5.

Figure 5. From top to bottom, it displays the problems of overhead lines, temperature, and both wire and temperature, respectively.

VII. Conclusion

In this study, a universal framework that is compatible with complex and large-scale power networks has been proposed. This framework not only adopts model-based approaches of situation awareness, but also data-driven models of SA, as a result of which, adequate cognition of electric power networks status is obtained. According to theoretical analysis, accidents caused by man-made operation mistakes and insufficient SA is reduced significantly by this framework. And it consists of cloud storage, generation of information system, situation awareness algorithms. Another formal definition of situation awareness in power systems is descripted in this paper. When it comes to the specific algorithms of situation awareness in this study, the Esper engine is used for processing complex events stream, which is only part of the situation awareness algorithm, instead of using decision trees or single method of CEP. Using the information of practical systems gathered from Esper engine as a single node, a logic tree is trained for representing information and achieving further situational awareness.

Tree structures offer advantages of fast inquiry, swift feedback, effective decision-making, and the ability to rapidly response adventitious developments. Insomuch as multi-granularity SA information can be extracted from the tree structure, it would be wiser to push corresponding fault warning information and decision-making information into system operators with different priorities. Construction method of storage also is elaborated in this study owing to the fact that the function of real-time access and inquiry of data system has important engineering significance. In the end, the technology of visualization is studied. The software development of this framework has already been completed and simple simulative experiments are carried out.

However, there are a few problems to be solved in this research. For one thing, in the absence of real datasets, it is impossible to achieve further cost evaluation and performance tests. In future work, key performance indicators of the system such as response time, reliability, and accuracy would be experimented and confirmed by using real data. For another, distributed deployment of this model would be accomplished in order to bring a better experience the users.

VII. Acknowledgement

Yu Huang is the corresponding author of this paper.

VIII. References

[1] Force U C P S O T, Abraham S, Dhaliwal H, et al. Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations, Natural Resources Canada, Ottawa[J]. 2004.

[2] Interim Report of the Investigation Committee on the 28 September 2003 Blackout in Italy, UCTE, 2003.

[3] Diao R, Vittal V, Logic N. Design of a Real-Time Security Assessment Tool for Situational Awareness Enhancement in Modern Power Systems[J]. IEEE Transactions on Power Systems, 2010, 25(2):957-965.

[4] Panteli M, Kirschen D S. Situation awareness in power systems: Theory, challenges and applications[J]. Electric Power Systems Research, 2015, 122:140-151.

[5] He X, Qiu R C, Ai Q, et al. Linear Eigenvalue Statistics: An Indicator Ensemble Design for Situation Awareness of Power Systems[J]. Statistics, 2015.

[6] Wehenkel L, Cutsem T V, Ribbens-Pavella M. An artificial intelligence framework for online transient stability assessment of power systems[J]. IEEE Transactions on Power Systems, 1989, 4(2):789-800.

[7] Sun K, Likhate S, Vittal V, et al. An Online Dynamic Security Assessment Scheme Using Phasor Measurements and Decision Trees[J]. IEEE Transactions on Power Systems, 2007, 22(4):1935-1943.

[8] Diao R, Vittal V, Logic N. Design of a Real-Time Security Assessment Tool for Situational Awareness Enhancement in Modern Power Systems[J]. IEEE Transactions on Power Systems, 2010, 25(2):957-965.

[9] Senroy N, Heydt G T, Vittal V. Decision Tree Assisted Controlled Islanding[J]. Power Systems IEEE Transactions on, 2006, 21(4):1790-1797.

[10] Balis B, Kowalewski B, Bubak M. Real-time Grid monitoring based on complex event processing [J]. Future Generation Computer Systems, 2011, 27(8):1103-1112.

I/O Performance Isolation Analysis and Optimization on Linux Containers

Li Zhou[1], Yifan Zhang[1], Youhuizi Li[1], Na Yun[1], Lifeng Yu[2]
[1]School of Computer Science, Hangzhou Dianzi University, Hangzhou, China
[2]Hithink RoyalFlush Information Network Co., Ltd.
huizi@hdu.edu.cn

***Abstract*—Container enables a new way to run applications by containerizing the application, which provides kinds of services to make them portable, extensible, and easy to be transferred between private data centers and public clouds. Comparing with virtual machines, containers have several advantages in terms of simplicity, low-overhead and light-weight. However, as the OS kernel and resources are shared by all the hosted containers, performance isolation becomes a challenging issue for guaranteeing their SLA.**

This paper discusses I/O performance isolation issue in container-based clusters. First, we analyze the characteristics of I/O performance isolation from the perspective of the SLA. Then we conduct the observation experiments using multiple containers to obtain the variation trend of the I/O performance parameters and observe the impact of the I/O overload container on the I/O performance isolation of the system. Finally, we propose two algorithms, SLAE and UTE, to improve the I/O performance isolation in container-based systems. These algorithms contribute to decrease the interference caused by the overloaded containers. Experimental results show the feasibility and effectiveness of our proposed algorithms.

Keywords-Container-based System; I/O Performance Isolation; SLA; Docker;

1. INTRODUCTION

Cloud service provides resources to users through the virtual machine [1]. Traditional virtual machine technology is based on the host hardware to build a virtualization management hypervisor, which allocates resources (CPU, memory, etc.) to each virtual machine [9], and each virtual machine has its own OS kernel. But this approach adds overhead when translates machine instructions from guest to host OS [5][14].

As the supporting platform that provides the microservice architecture for software applications, the container has attracted plenty of research attentions in the fields of cloud computing and virtualization. Containers, as Docker standardizes applications and services, opens another door for the operating environment of the application services. Compared with the traditional virtual machine, container isolates resources and permissions into a sandbox, which called a container, and each container is a separate resource usage space. Container isolation technology comes from LXC [7], it achieves environmental resources isolation by the namespaces mechanism and restricts the use of resources by the Cgroups mechanism. Namespaces mechanism and Cgroups mechanism are provided by the Linux kernel. However, because all containers at the same host share one OS core and multiple containers running simultaneously, there will be inevitably disturbances between containers. To improve the performance of container, isolation and safety are the key issues that need to be studied and solved.

The rest of this paper is organized as follows. We introduce the background information and analyze the characteristics of I/O performance isolation in Chapter 2. Chapter 3 shows the variation trends of I/O performance parameters and argues the influence of the overload container on the I/O performance isolation. Chapter 4 describes a prototype system that can dynamically maintains I/O performance isolation of the system and illustrates two different maintenance algorithms, then verifies the feasibility of algorithms. Chapter 5 concludes the paper and discusses future works.

2. BACKGROUND

2.1 Containers

The container technology isolates resources and permissions into containers. Docker makes isolation for PID, UTS, IPC, Network and other environmental resources through the NameSpaces mechanism provided by Linux kernel, besides, it limits the use of CPU, Memory, I/O and other shared resources by leveraging the Cgroups mechanism. These constraints can allow containers follow certain rules at runtime.

Docker provides a few commands which use the Blkio in Cgroups mechanism to restrict containers' I/O usage. There are three groups of I/O configurations available in Docker so far. The first group is: --blkio-weight, which can set the I/O weight for a container. The second group is: --device-write/read-bps, which can limit the read/write rate (bytes per second) from/to a device. The last group is: --device write/read-iops, which can limit read/write rate (I/O per second) from/to a device for a container.

Although Docker provides several ways to restrict the use of the container's I/O. But it requires the admin estimate the I/O usage of the container first, and then admin use the above methods to restrict the I/O of the container. This kind of method is obvious hysteresis and uncertainty, especially when I/O competition becomes intense and complex. Current container management software such as the popular Kubernetes, a tool for scheduling and managing containers [7].

DOI reference number: 10.18293/SEKE2017-201

The smallest unit for managing of Kubernetes is a Pod, which is a set of containers, Kubernetes is responsible for completing their management. The main purpose of the container-based cloud management software is to cluster the deployment and management of containers. However, there is no specific guarantee for the isolation between containers on the host OS.

Sean McDaniel et al. [3] proposed a two-tiered approach to guarantee I/O quality of service in Docker containers. They thought that combining the node-level QOS and the cluster-level QOS together will have better load balancing and higher resource utilization. Miguel Xavier et al. [2] found the container-based system is not yet mature to ensure performance isolation among disk-intensive workloads. To address this problem, they [2] proposed to combine different types of loads on containers to reduce the interaction. Their results suggest the combination of CPU intensive and I/O intensive to alleviate the performance impacts, rather than a combination of disk-intensive and memory-intensive.

2.2 Isolation

2.2.1 I/O Performance Isolation based on SLA

In general, cloud providers are responsible for guarantee the SLA for their users. They will allocate the number of containers and the resource cap in order to ensure that the SLA requirement of all the containers on the host can be satisfied at the same time. But if some users or applications produce excessive load, it is possible that the balance is destroyed and other containers' performance is influenced.

Based on the SLA, we can classify the I/O performance isolation of container-based system into two categories as follows:

Poor I/O Performance Isolation: When the multi-container running on the host, the I/O interference between containers is very strong, and the I/O state of the container is easy to change. Especially when some containers produce overload I/O operations, it will cause other containers' I/O performance fail to reach the SLA requirement, such as I/O wait time and IOPS. In this case we believe that the I/O performance isolation of the container-based system is poor.

Excellent I/O performance isolation: When the multi-container running on the host, the I/O interference between containers is very weak, and the I/O state of the container is very stable. Even if some containers produce overload I/O operations, because the system has a good I/O performance isolation, it will still guarantee other containers' I/O performance reach the SLA standard. In this case we believe that the I/O performance isolation of the container-based system is excellent.

2.2.2 WorkLoad Model

We can treat containers as processes that produce I/O, since they share one OS kernel, we can assume the I/O queue is a shared service queue. So the system can be viewed as the M/M/1 model that have single-queue and single-server.

In the M/M/1 model of single-queue and single-server, we know W = S/(1-U) according to Litte's Law and Utilization Law [4]. Where W is the job wait time, U is the utilization of the system, and S is the actual service time of the job. Based on the equation, we can observe that the higher current disk device I/O utilization is, the greater the I/O delay will be, and when the utilization exceeds a certain threshold or so, the I/O wait time will appear rapid growth.

3. Performance Trends And Impact Analysis

3.1 Experimental Setup

3.1.1 Experimental Environment

We use a node server as a host, the node has 16 AMD Opteron(TM) Processor 6136 CPU and 32GB of Memory capacity, and it equipped with CentOS7 and Docker 1.12.5.

3.1.2 Experimental Content

We execute the script file in the container to generate the I/O request to write the file. The rate of script writing is relatively small, so we use the Linux DD command to simulate the overload I/O case. It is worth noting that the I/O should avoid Buffer for the accuracy of the observation.

The observation is mainly divided into two parts. The first part we open a number of containers on the host while running the script file to generate write I/O, the variable is the number of growing containers, meanwhile observing the trend of the I/O indicators. The second part of the experiment, we run a number of normal containers coupled with a container generate high I/O load to observe the effect of overload container on the I/O performance for other normal containers. Since each container has a specific disk device number, so we observe the I/O of the corresponding container through the disk device.

3.2 Results And Analysis

3.2.1 IOPS, write rates show a decreasing trend of power function

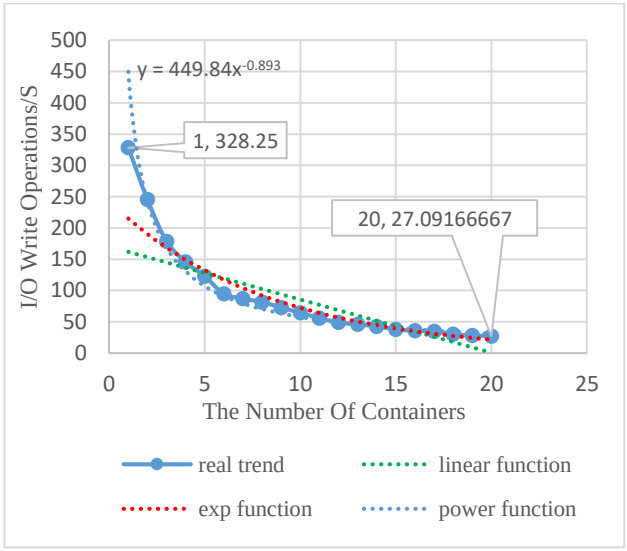


Figure 1. Variation trend of I/O write operations/s

We allow each container to achieve the same I/O load. As we can see from Figure 1, with the number of containers increased from 1 to 20, the average I/O write operations is reduced from 328 times per second to 27 times per second. The real data trend is a solid line, the whole process fits closely

the decline of the power function, rather than fits other functions.

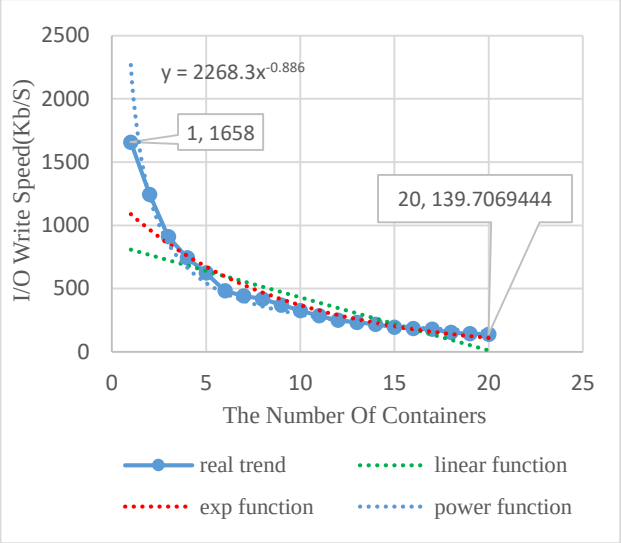


Figure 2. Variation trend of I/O write speed

Similarly, in the above case, we can see from Figure 2, the average write speed of each container also shows a similar exponential decline trend. On our machine, the write speed decreased from 1658Kb/s to 139Kb/s. The real data trend is the solid line and the fitting curve is represented by the dotted line.

3.2.2 I/O utilization, the length of wait queue fit logarithmic function growth trend

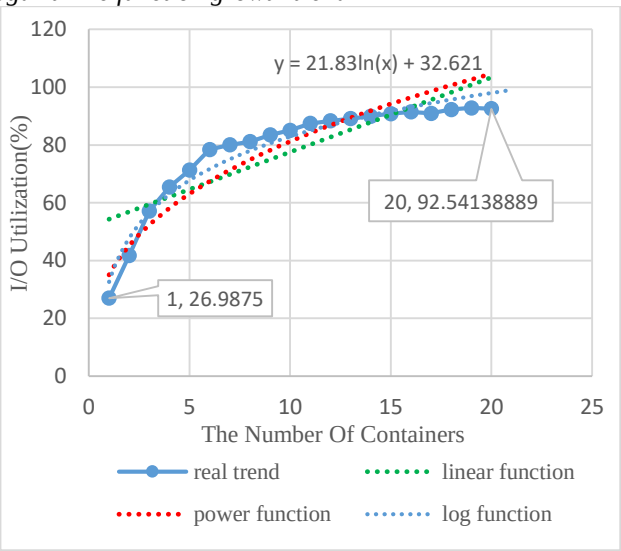


Figure 3. Variation trend of I/O utilization

From Figure 3 we can see that with the number of containers increased from 1 to 20, the average I/O utilization of disk device increased from 26.98% to 92%. The variation trend of real data is the solid line, we fit the whole process very close to the growth of the logarithmic function, rather than other functions.

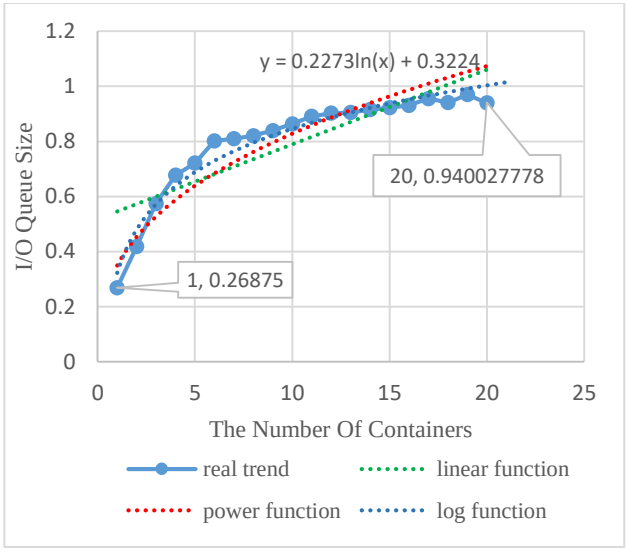


Figure 4. Variation trend of I/O queue size

Similarly, in the above case, we can see from Figure 4, the waiting queue length of containers is also growth as the logarithmic function. On our machine, the length of the I/O queue increased from 0.268 to 0.94. The real variation trend of the data is the solid line and the fitting curve in the graph with the dotted line.

3.2.3 The I/O utilization exceeds a certain threshold, then wait time soared

We increase I/O utilization of the system by increasing the number of containers, and observe the relationship between the I/O utilization and I/O wait time.

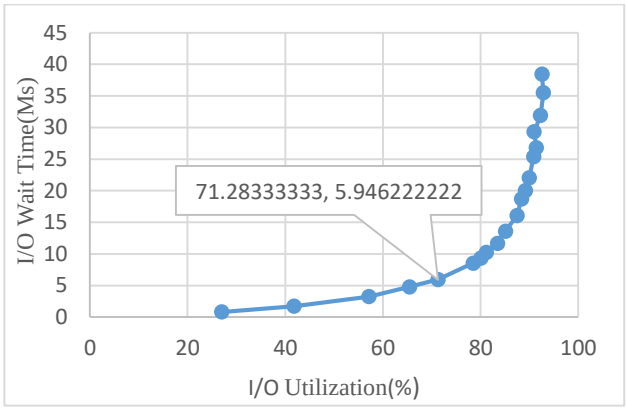


Figure 5. Variation trend of I/O utilization and wait time

As Figure 5 shows, with the increase of containers, the I/O utilization increases from 27% to 92%. And the wait time is close to a linear growth before the I/O utilization reaches 70%-80%. When the I/O Utilization is 71%, the average I/O wait time is about 5.9Ms. However, when the I/O utilization exceeds 70%-80%, the wait time grows rapidly near the exponential function. Soon the wait time is up to 30Ms or more, which is far exceeded the upper limit of SLA for a normal disk I/O.

3.2.4 Overloaded container will cause damage to the I/O performance isolation of the system

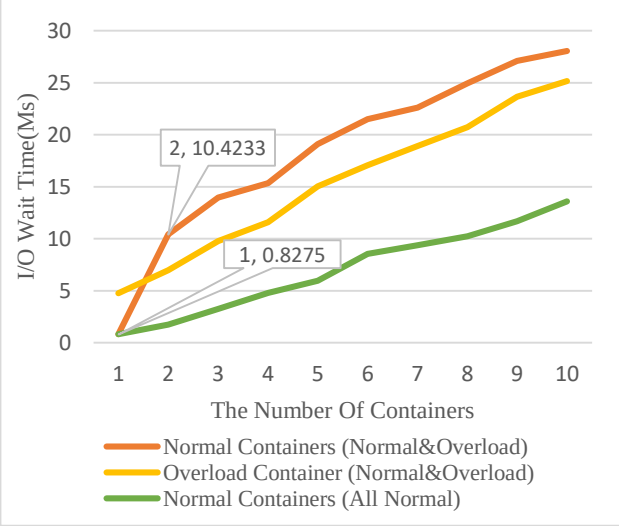


Figure 6. Variation trend of I/O wait time when all normal containers and one overloaded container & some normal containers

Figure 6 describes the impact of a container with a high I/O utilization on the performance of other normal containers. If the I/O loads of all hosted containers are normal, with the increase of containers, we can see the variation trend of I/O wait time fits a slow growth of a linear. When running 1-9 normal containers with an overloaded container, the result shows that the overload container resulted in a significant increase in the I/O wait time of the normal containers, while the change in the overload container is small.

Here we define ΔW = W_New – W_Pre, which represents the change in the I/O wait time of one container. W_New indicates the I/O wait time of the container before the overload container appears, W_Pre indicates the I/O wait time of the container after the overload container appears, it can reflect the impact of container's I/O performance. The large ΔW shows big impact, the small ΔW shows small impact. And we use ΔSW to indicate the size of I/O performance isolation of the system that has m containers when the I/O strength of the container n grows. The more containers the system has, the more susceptible the system's I/O isolation is. The smaller ΔSW is, the more excellent the system isolation is.

$$\Delta SW = \sum_{\substack{0 \le i \le m \\ i!=n}} \Delta Wi$$

When there is only one normal container, the I/O wait time is approximately 0.82Ms. After an overloaded container appears, the I/O wait time of the normal containers rises immediately to 10.4Ms, at this time ΔM = 10.94-0.82 = 10.1Ms, while the I/O wait time of overload container rises from 4.7Ms (running alone) to 6.9Ms, the ΔM is only 2.2Ms. With the increase of containers, the ΔM of each container changes only a little, while the ΔSM is growing fast because of the increase in the number of containers.

3.3 Summary

In order to gain a deeper understanding of the trend in the I/O performance and the intrinsic relationship between the I/O parameters in the container system, we summarized some of the above meaningful conclusions as follow. We found that as the number of containers grows, IOPS, write rates show a decreasing trend of the power function. The reason why they are not a linear is due to the presence of I/O kernel scheduling. Meanwhile, we found that I/O utilization and the length of wait queue present logarithmic function growth trend. Then we run normal containers and add an overloaded container at the same time. The observation parameter is the I/O wait time which is universality, we find that the emergence of overload container led to significantly waiting time increasing of other containers, the I/O performance isolation of system is damaged. As the number of normal containers increases, ΔW itself does not change much, but the value of ΔSW is rising. At the same time, we found that the ΔW value of the overload container is smaller relative to the ΔW value of the normal container. We propose that the I/O strength of the overload container should be reduced to maintain I/O performance isolation of container-based system. And the container with the highest I/O utilization and read/write speed should be regulated first for both fairness and effect priority.

4. System And Algorithms Design

4.1 System Design

According to the observations, we found that the I/O performance isolation of the container-based system is not guaranteed, especially when overload containers exist, it may cause strong interference to other containers. So we propose a system which can dynamically regulates containers' I/O to provide I/O performance isolation and I/O load balancing for the container-based system.

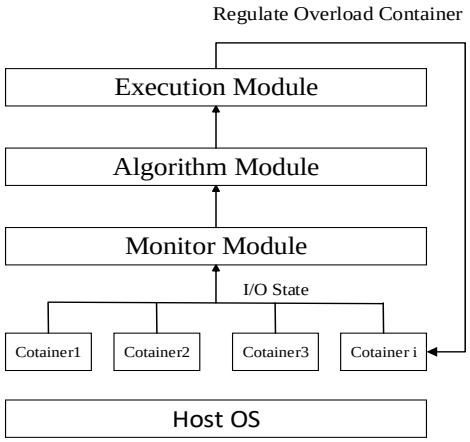


Figure 7. I/O dynamic adjustment system framework

Figure 7 demonstrates the architecture of the system. It is built on the host OS and composed of 3 major modules: Monitor Module, Algorithm Module, and Execution Module.

Monitor Module: The module is responsible for collecting real-time I/O status of all containers running on the host currently, including IOPS, read/write rate, I/O wait time, the average length of the request queue, I/O utilization etc. It will pass this information to the Algorithm Module.

Algorithm Module: The module is responsible for policy analysis based on the information collected by Monitor

Module. Adjustment algorithms determine whether the I/O performance isolation of the system will be destroyed. Based on the algorithm, this module decides if it is necessary to regulate the containers' I/O, and the corresponding results are sent to the execution module.

Execution Module: If the result of the algorithm module is that the I/O performance isolation of the current system needs to be maintained, then the execution module will perform the specific operation accordingly and use Cgroups to restrict the I/O intensity of the overload container appropriately so that the I/O performance isolation of the container-based system is maintained.

4.2 Algorithms Design

According to the results of analysis and observations, we designed the following two dynamic adjustment algorithms for judging the I/O performance isolation.

4.2.1 SLAE(Service-Level Agreement Ensure) dynamic adjustment algorithm

In the previous section, we discussed the I/O performance isolation of container-based system based on the SLA. And we have experimentally observed that when an overloaded container generates excessive I/O requests, it may cause I/O state of other containers fail to meet SLA requirements.

In the SLAE dynamic adjustment algorithm, we use the SLA requirements of the I/O service as the parameter to evaluate the isolated damage. Take I/O wait time as an example. If the overloaded container has caused the I/O Wait Time of other normal containers exceed the SLA requirements, we believe that the I/O performance isolation of the current container-based system has been disrupted. Hence, we decrease the overall I/O utilization of the system by reducing the I/O strength of the overload container so that the I/O wait Time, as well as other parameters of other containers, can be guaranteed, that is, the I/O performance isolation of the system is guaranteed.

4.2.2 UTE(Utilization Threshold Ensure) dynamic adjustment algorithm

Since the multi-container case can be seen as a model of single-queue and single-server. And experiments verify that when I/O utilization of the system exceeds a threshold, the average I/O wait time of the container will be a sharp increase, then I/O state will be very unstable, which means I/O performance isolation is poor.

So judging I/O performance isolation of the system can also depend on the I/O utilization of the system. We can set a threshold to the I/O utilization of the system. And the threshold is about 0.7-0.8 according to the observation results. We know that when the I/O utilization exceeds the threshold, the I/O wait time of all containers becomes high and the rate of increase is accelerating, hence the I/O performance isolation of system is threatened. At this time, the execution module chooses the container with the highest I/O utilization in the current running containers and reduces its I/O intensity, so that the I/O utilization of the system is reduced, the I/O performance isolation and load balancing of the system are maintained.

4.2.3 Algorithms comparison

When the service provider and the user have signed the SLA, the SLAE algorithm should be adopted. If users bought the service, then the enterprise should ensure the service quality. For users, if the I/O metrics reach the SLA requirements, then the I/O service performance is good. Therefore, if there is a clear SLA requirement, it is recommended to use the SLAE algorithm.

In the absence of SLA situation, for example, the container-based system is used to be the operating environment of applications. The operating conditions of application are complex and unstable. To ensure I/O load balancing, the UTE algorithm should be used. Our experiments show that when the I/O utilization exceeds a certain threshold, the I/O performance of the applications in containers will be unstable and drops rapidly. The UTE algorithm can guarantee I/O performance and load balancing for container applications in this situation.

4.3 Regulation of Overload Container

Linux's Cgroups mechanism can restrict the I/O strength of a particular process on a specific device. Each container has a main process at the host, all application processes within the container are child processes of the main process. Docker has assigned a separate disk device for each container, each of disk devices has its own disk device number. So we can set the Cgroups group by the specific device number and process ID, reduce the corresponding container process for their own disk device I/O strength to achieve the corresponding restriction of the container's I/O.

We run three normal I/O containers and a strong overload I/O container simultaneously. Then we restricted the I/O of the overload container and study the changes of the I/O wait time of the three normal containers.

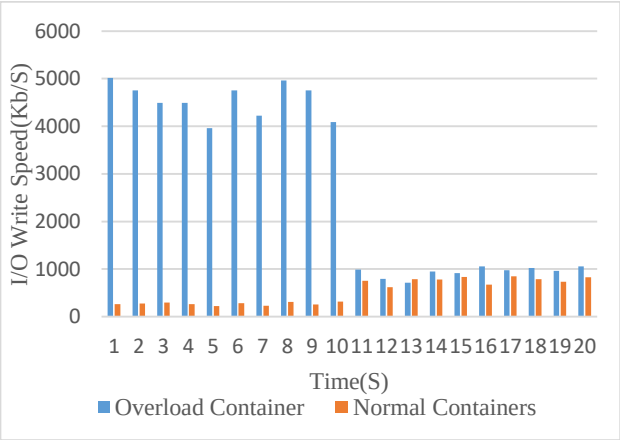


Figure 8. The I/O write speed's variation of containers

Figure 8 depicts the variation in I/O write speed of the overload container and the normal containers. We can see that in the first 10 seconds, the I/O write speed of the overload container is 4.5Mb/S or so, then it immediately reduced to about 1 MB/s due to our restriction, and the I/O write speed of the normal container raised simultaneously.

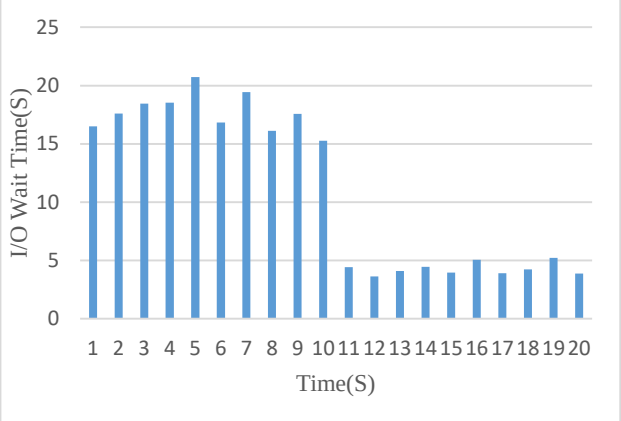


Figure 9. Variation of normal containers' I/O wait time

Figure 9 describes the variation of the average I/O waiting time. In the first 10 seconds, the average I/O wait time of normal containers reached 17.7Ms, and the maximum has exceeded 20Ms due to the high I/O write speed and high I/O utilization of the overload container. After 10 seconds we restrict the I/O write speed of the overload container, the I/O utilization of the system is reduced, we can see the average I/O wait time of normal containers drops immediately below 5Ms which is an excellent level.

4.4 Summary

In order to improve the I/O performance isolation of the container-based system, we propose a system which can dynamically maintain the I/O performance isolation. It consists of three modules, Monitor Module, Algorithm Module and Execution Module. We design two judgment algorithms, the SLAE algorithm and the UTE algorithm, and we discuss their details as well as usage scenarios. In order to verify the feasibility of the algorithms, we did the following experiments. We run an overloaded container and three normal containers on one host at the same time. We observe that the overload container greatly influenced on the I/O performance of other normal containers. We use the Cgroups mechanism to restrict the I/O intensity of the overload container. The results show that, we restrict the overload container's I/O write rate from 4.5Mb/s to 1Mb/s, so the average I/O wait time of the normal containers is reduced from 17.7Ms to below 5Ms. The I/O performance isolation of the container-based system is guaranteed, and the feasibility of the algorithms is also proved.

5. Conclusion And Future Works

The performance isolation is an important issue for the container-based system. In this paper, we analyzed the characteristics of I/O performance isolation from the perspective of the SLA. We observed the variation trend of containers' I/O performance parameters under the condition that multiple containers running simultaneously, and discussed the impact of the I/O overload container on the I/O performance isolation of the container-based system. In order to improve the I/O performance isolation of container system, we proposed a system that can dynamically maintain the I/O performance isolation. Besides, we designed two I/O isolation maintenance algorithms and discussed their usage scenarios. Moreover, we use Cgroups mechanism to restrict the I/O strength of the overload container in order to reduce the impact on other normal containers, so the I/O performance isolation of the container-based system can be guaranteed, and the feasibility of the algorithms is also proved.

Our future research is to achieve dynamic maintenance of I/O isolation systems, and design more algorithms to further improve the I/O isolation performance. Besides, in order to comprehensively enhance the performance isolation of the container-based system, we plan to extend the above ideas to other shared resources such as CPU and Memory etc.

Acknowledgment

This work is supported by the National Key Technology R&D Program under Grant (No.2015BAH17F02), the NSF of China (No. 61572163) and the NSF of China (No. 61602137).

References

[1] R. Krebs, C. Momm, and S. Kounev, "Metrics and techniques for quantifying performance isolation in cloud environments," Science of Computer Programming, 2012.

[2] M. G. Xavier, I. C. Olivera, F. D. Rossi, and R. D. D. Passos, "A performance isolation analysis of disk-intensive workloads on container-based clouds," Euromicro International Conference on Parallel, 2015, pp. 253-260.

[3] S. Mcdaniel, S. Herbein, and M. Taufer, "A two-tiered approach to I/O quality of service in docker containers," IEEE International Conference on Cluster Computing, 2015, pp. 490-491.

[4] P. J. Denning, and J. P. Buzen, "The operational analysis of queueing network models," Computing Surveys, vol. 10, no. 3, pp. 225-261, 1978.

[5] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," Lecture Notes in Computer Science, 2014, pp. 438-453.

[6] R. Peinl, F. Holzschuher, and F. Pfitzer. "Docker cluster management for the cloud - survey results and own solution." Journal of Grid Computing, vol. 14, no. 2, pp. 1-18, 2016.

[7] D. Bernstein, "Containers and cloud: from LXC to docker to kubernetes," IEEE Cloud Computing, vol. 1, no. 3, pp. 81-84, 2014.

[8] C. Boettiger, "An introduction to docker for reproducible research," Acm Sigops Operating Systems Review, vol. 49, no. 1, pp. 71-79, 2015.

[9] A. M. Joy, "Performance comparison between linux containers and virtual machines," Computer Engineering and Applications IEEE, 2015, pp. 342-346.

[10] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," Linux Journal, vol. 2014, no. 239, 2014.

[11] K. T. Seo, H. S. Hwang, I. Y. Moon, O. Y. Kwon, and B. J. Kim, "Performance comparison analysis of linux container and virtual machine for building cloud," Networking and Communication, 2014, pp. 105-111.

[12] J. K. Patel, S. Akhtar, V. K. Agrawal, K. N. Bala, S. Murthy, and A. R. Manu, "Docker container security via heuristics-based multilateral security-conceptual and pragmatic study," IEEE–Iccpct, 2016.

[13] A. Celesti, D. Mulfari, M. Fazio, M. Villari, and A. Puliafito, "Exploring container virtualization in IoT clouds," IEEE International Conference on Smart Computing, 2016, pp. 1-6.

[14] R. Dua, A. R. Raja, and D. Kakadia, "Virtualization vs containerization to support paas," IEEE International Conference on Cloud Engineering, 2014, pp. 610-614.

[15] N. Kratzke, "About microservices, containers and their underestimated impact on network performance," Cloud Computing, 2015, pp. 165-169.

A Framework to Build Bayesian Networks to Assess Scrum-based Software Development Methods

Mirko Perkusich*[1], Kyller Gorgonio†[1], Hyggo Almeida‡[1], and Angelo Perkusich§[1]

[1]Embedded and Pervasive Computing Laboratory, Federal University of Campina Grande, Campina Grande, Brazil

Abstract

Agile software development has been increasingly used to satisfy the need to respond to fast moving market demand and gain market share. Scrum, which is a project management framework, dominates as the most popular agile method. In the literature, there are a number of solutions to customize and assess Scrum-based agile methods, but they are limited to focus only on process factors, assume a predefined set of practices or rely only on subjective evaluation. This paper presents a framework to build a Bayesian Network to assist on the assessment of Scrum-based software development methods. The BN models the main entities of the software development process and can be complemented with software practices and metrics. To evaluate the completeness of our solution, we performed simulations to check if the proposed framework diagnoses 14 known Scrum anti-patterns extracted from the literature. 12 anti-patterns were directly detected, 1 was indirectly detected by the BN and 1 was considered as invalid. We concluded that the proposed solution is complete to detect the major flaws of Scrum-based software development methods and can be used to assist on the configuration, adoption and continuous improvement of Scrum teams.

Agile Methods; Bayesian Network; Software Metrics; Scrum; Method Engineering

*mirko.perkusich@embedded.ufcg.edu.br
†kyller@embedded.ufcg.edu.br
‡hyggo@embedded.ufcg.edu.br
§perkusic@embedded.ufcg.edu.br

1 Introduction

Agile Software Development (ASD) methods have been increasingly used to satisfy the need to respond to fast moving market demands and gain market share. In contrast with traditional plan-driven processes, agile methods focus on people, are communication-oriented, flexible, fast, light, responsive and oriented to learning and continuous improvement [7]. As a consequence, subjective factors such as collaboration, communication and self-organization are key to evaluate the maturity of agile software development.

Scrum dominates as the most popular agile process. It is a software project management framework and must be complemented with technical and managerial practices and processes to define a software development method. If not complemented properly, Scrum might result in reduced productivity and product quality [23, 24].

The method definition depends on customization factors such as team characteristics, internal environment, external environment, project goals, maturity level and previous knowledge [2]. In the context of ASD, most solutions to assist on agile methods configuration are based on Method Engineering [1], in which the team configures a method adapting existing methods (e.g., Rational Unified Process, Scrum and Extreme Programming).

To adopt agile methods, some studies propose combining ASD with CMMI [17]. Others, are specific to ASD [2, 16, 24, 8, 10]. The proposed solutions have one or more of the following limitations: focuses only on the process, assume a predefined set of practices or rely only on subjective data.

Given this, we conclude that there is no consolidated approach to assist on the adoption and continuous improvement of agile methods. In this paper, we present a framework to build a Bayesian Network (BN) to assist on the assessment of Scrum-based software development methods. The BN models the main entities of the software develop-

DOI reference number: 10.18293/SEKE2017-139

ment process and can be complemented with software practices and metrics. With the resulting BN, it is possible to monitor if the agile method is being followed and if it meets the needs of the project to assist on decision-making.

Bayesian networks are probabilistic graph models and used to represent knowledge about an uncertain domain. A Bayesian network, B, is a directed acyclic graph that represents a joint probability distribution over a set of random variables V. The network is defined by the pair $B = \{G, \Theta\}$. G is the directed acyclic graph in which the nodes $X_1, \ldots, X_n$ represent random variables and the arcs represent the direct dependencies between these variables.

The problem of modeling the Scrum process involves causal reasoning and aleatory uncertainty (i.e., intrinsic variability), which can be reduced by gathering more data or eliciting knowledge from experts. Given that the framework must be used by practitioners, the inferences must be clear. Furthermore, the models must be adaptable and assist on decision-making. According to Verbert et al. [20], Bayesian network is a suitable approach for this context.

This paper is organized as follows. Section 2 presents the proposed solution. Section 3 presents the results of the validation; and Section 4 presents our conclusions, limitations and future works.

2 Framework description

The goal of the framework is to build a BN to model a Scrum-based software development method. Scrum Masters should use it to guide the configuration of the agile method and to assess it. As a result, the team has an instrument to support its continuous improvement. As input, the BN receives data collected from the Scrum Master (e.g., through a questionnaire) or automatically through tools. The BN outputs data with probability values that represent the current status of key factors in the software development method. The Scrum team should use the data to, during Sprint Retrospective meetings, diagnose problems and prioritize the areas that must be improved. The data should support discussions regarding action points to be executed to improve the quality of the software development method and, consequently, the project's chances of success. The discussions should be a collaborative activity and involve the stakeholders responsible for decision making such as the Product Owner, Scrum Master and development team. Its usage should be supported by a process such as the one presented in Perkusich et al. [14].

The framework consists of a BN that models the main entities of a software development method based on Scrum. It is composed of two types of fragments: frozen and hot. The frozen fragment is related to official artifacts, resources and practices of Scrum as presented in the Scrum Guide [19]. It should not be modified, unless there are plausible justifications based on domain expert knowledge

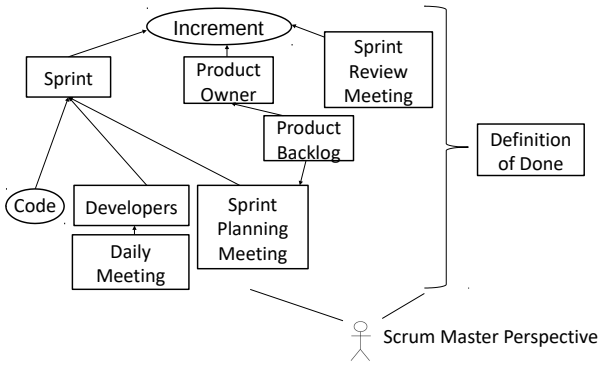


Figure 1. Overview of the framework.

or historical data. The frozen fragment defines the base BN. The reasoning behind having a frozen fragment is that even though some deviations of the Scrum Guide may be well motivated and reasonable, teams are tempted to adjust Scrum for the company without clearly understanding the consequences of the deviations as evidenced by Eloranta et al. [4]. For instance, not having an ordered Product Backlog might happen in the context of having a Product Owner without authority, a Product Owner that is not part of the Scrum team, lack of feedback loops (e.g. Sprint Review meetings) or an incompetent Product Owner [4]. As a consequence, the team might build wrong features and avoid features which are hard to implement or test, increasing the risk of problems arising in the late stages of development. On the other hand, in the context in which features are fixed and given upfront, it is reasonable to not maintain an ordered backlog. Therefore, given that the modifications on the frozen fragment mean deviation of Scrum recommendation, they must only occur if the consequences are understood and reasonable; and not to adapt Scrum to the company's current *status quo*.

Hot fragments are related with parts of the BN that models practices and processes that should complement the Scrum framework. Furthermore, it is possible to complement the model with fragments representing software metrics-based models (i.e., set of metrics defined for a given purpose). For instance, Quamoco [22], which focuses on measuring product quality, can be associate to the BN. The hot fragments must be defined given the context of the project and must be defined according to the project's Definition of Done.

An overview of the framework is shown in Figure 1, in which the rectangles represent fragments (i.e., set of nodes) and the ellipses represent nodes in the Bayesian network. Notice that the Bayesian network is defined from the viewpoint of the Scrum Master and it must be used to assist during Sprint Retrospective meetings. Therefore, the Scrum Master, a role in Scrum, and the Sprint Retrospective meet-

ing, a Scrum practice, are not represented. Furthermore, the Definition of Done, a Scrum practice, for being an umbrella practice is also not represented.

The problem of building a BN can be divided into: (i) construct the Directed Acyclic Graph (DAG) and (ii) define the Node Probability Tables (NPTs). In Section 2.1, we present details of constructing the DAG of the base BN (i.e. frozen fragment). In Section 2.2, we show the process of complementing the DAG of the base BN with practices and metrics (i.e., hot fragments). The constructed BN is based only on ranked nodes [6], which are based on an ordinal scale. Regarding the definition of the NPTs, this paper is limited to state that collected data from a domain expert and applied the process presented in Fenton et al. [6].

2.1 Base BN construction

The problem of constructing a DAG can be subdivided into: (i) defining the nodes and edges and (ii) defining the scale (i.e., states) of the nodes and the associated probabilities. In the context of our work, a node represents a process entity and there is an edge whenever the entities relate to each other. To solve the second subproblem, we, as other tools to evaluate agile methods such as ComparativeAgility [24], used a five points Likert (i.e., ordinal) scale. Therefore, for each node (i.e., process factor), five possible states were defined. For metrics with numerical scale, thresholds must be defined given the context of the project to map the numerical scale to an ordinal one. The thresholds can be defined using a domain expert knowledge or analysis of historical data.

To solve the first subproblem, we used an incremental approach. Previous versions of the BN were presented in Perkusich et al. [14] and Perkusich et al. [13]. In this paper, we present the most recent version. In comparison with past versions of the base BN, the differences are that (i) we added the concept of frozen and hot fragments, (ii) reduced the size of the base BN from 73 to 44 nodes and (iii) improved the modeling of team (i.e., human) factors.

We followed a top-down approach in which we decomposed the top-level node into factors (i.e., process entities) that we judged to be observable by a Scrum Master or collected through tools. Given that the main goal of agile software development is to satisfy customers with working code and Scrum is an incremental approach, we defined *Increment* as the top-level node. We decomposed it using a well-known DAG idiom: synthesis [5]. Due to space constraints, we only show how we built part of the base BN.

In Scrum, the increment is developed during sprints. The Product Owner is responsible for maximizing the value of the product. Furthermore, the increment must be evaluated during Sprint Review meetings. Therefore, we added the nodes *Sprint*, *Product Owner* and *Review meeting* as parents of the node *Increment*.

Table 1. Examples of practices that can be associated with nodes of the base BN.

Node	Practices
Code	Refactoring, pair programming, Test-Driven Development, Continuous Integration, peer code review and test automation.
Estimation	Planning Poker, big wall, spike [23], story point, ideal hours and T-shirt size.
Product Backlog	Grooming meeting, Release plan [15] and product vision [15].
Ordering	Kano model, Wieger model, OR value, Innovation games, Return of Investment (ROI) and MoSCoW.
Details	User story and Use case
Planning meeting	Sustainable pace [23], Stabilization [23] and Velocity-driven.
Delivery plan	Interaction Room [9].
Monitoring	Burndown and Burnup [19].

During Sprint Review meetings, the product must be inspected and, if necessary, adapted by the customers. As previously explained, not using this practice or not having the customers participating might result in rework and implementation of wrong features. Therefore, we added the nodes *Inspection* and *Adaptation* as parents of the node *Review meeting*. To input data into the model, the Scrum Master must observe the Sprint Review meetings and judge these two factors. As shown in Section 2.2, it is also possible to complement this fragment with metrics to indicate the evidence of these input nodes.

In past evaluations with industry practitioners [12, 14], we noticed that, in practice, it is common for the customer to not be available during all Speint Review meetings. On the other hand, the client evaluated the increment on other informal meetings. Therefore, given that the feedback loop is short and there is a reasonable justification, this is a case in which a frozen spot on the base BN could be modified. An auxiliary node can be added to model this behavior and compensate the lack of participation of the customer during the Sprint Review meetings. The complete DAG for the base BN is shown in Figure 2.

2.2 Complementing the BN with practices and metrics

Given that Scrum must be complemented with practices and processes, the BN can be improved to suit the needs of the given project. Each node in the BN can be mapped to a set of practices. The set of practices is represented as an

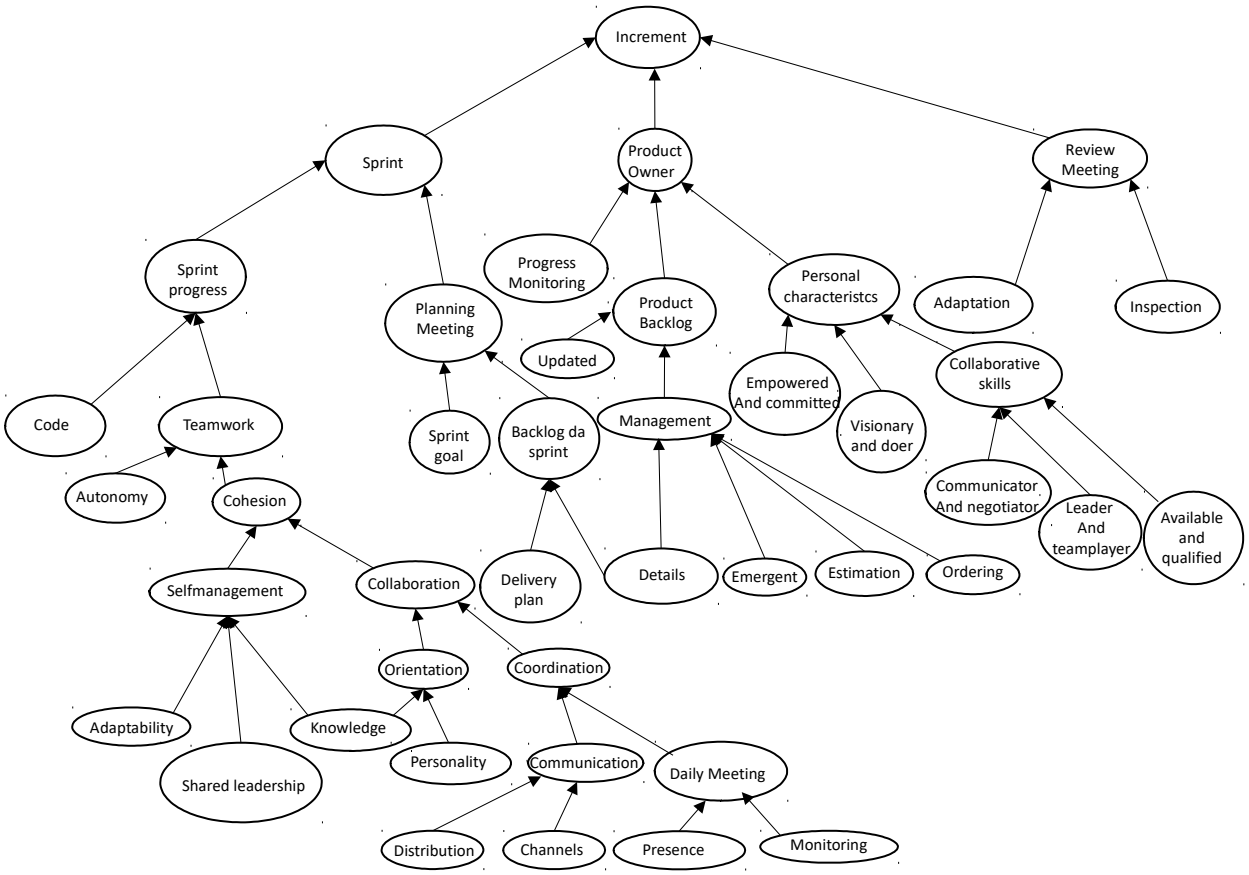


Figure 2. DAG of the base BN.

auxiliary node in the BN. The node is a copy of the given factor. For instance, we can map the node *Code*, to *Node-practices*. Therefore, the evidences on *Code* will indicate the quality of the code production process.

Despite its simplicity, this procedure assists the team on analyzing the BN during the Sprint Retrospective meeting. Given evidences collected during the sprint, the team identifies which processes and practices needs to be modified to increase the chances of building products that satisfy the customers. For this purpose, it is important that the team registers which practices or processes are being used for each factor. In Table 1, we show examples of practices that might be associated with nodes of the base BN.

Another option is to decompose the auxiliary using DAG construction techniques into a causal model representing the given process. Even though it increases the cost of constructing and maintaining the BN, it enables a deeper analysis of the process. For instance, according to Pichler [15], defining a release plan and product vision is a good practice to plan agile projects and influences the quality of the product backlog. The release plan is composed of the release date, description of top features and product vision. The product vision must have a broad and engaging goal, be clear and stable, short and concise and describe the critical attributes to satisfy the needs of the customers. With this information, we can build the causal model shown in Figure 3 and associate it with the node *Product backlog* of the base BN.

Furthermore, it is possible to complement the base BN with metrics. Metrics can be used with two purposes: (i) as indicator of values for input nodes and (ii) warnings. For (i), it is necessary to connect the metrics to the given node and calibrate the NPT. If the metric is automatically collected through tools, this approach can reduce the cost of using the BN. For instance, we could use static analysis metrics to indicate the quality of the code. On the other hand, manually collected metrics can also be added. For instance, to indicate if the Product Backlog is ordered correctly, we can evaluate the technical dependencies, risks and business value [21]. The resulting fragment is shown in Figure 4.

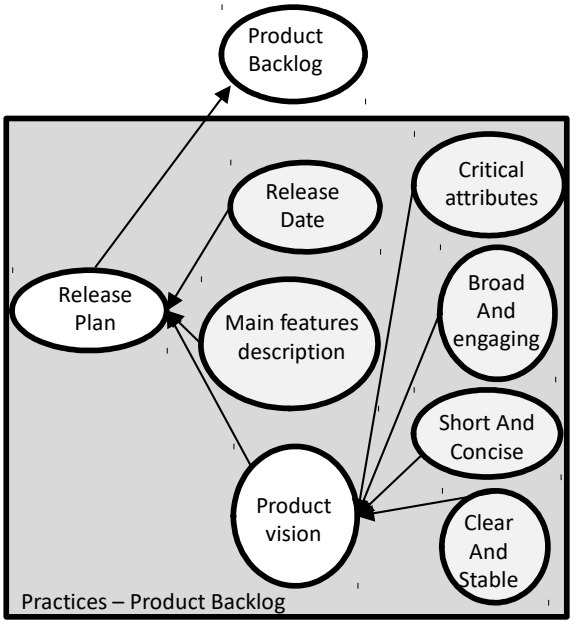


Figure 3. Example of decomposing an auxiliary node.

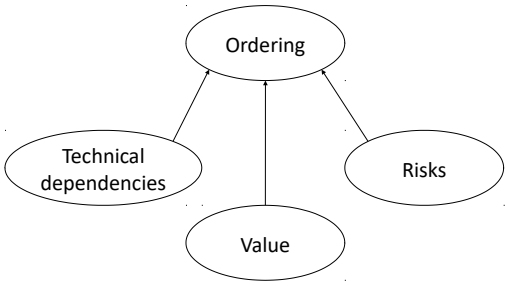


Figure 4. Example of adding metrics to the base BN.

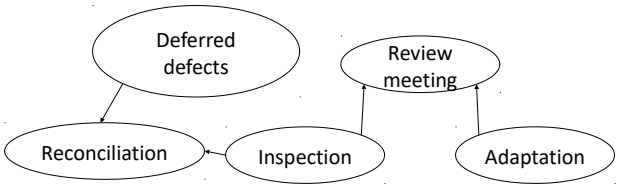


Figure 5. Example of adding warning metric to the base BN.

For (ii), the metric is a warning that indicates that something is wrong with the BN. Furthermore, it helps to identify possible problems in the software development method that the BN failed to identify, which might be a result of problems with the construction, usage or modeling limitation. In this case, the metric should be connected to the base BN using the reconciliation idiom [5].

For instance, we can associate the metric *Deferred defects* [11] to the node *Inspection*. This metric indicates the number of defects that should have been detected during the previous sprint, but were not. Suppose the Scrum Master judges that the Sprint Review meetings for a given project are being executed satisfactorily, but *Deferred defects* were identified. This might raise a warning meaning that increments with defects were approved during Sprint Review meetings, which means that the inspection was not as satisfactorily as judged by the Scrum Master. The resulting fragment for this example is shown in Figure 5. On the other hand, depending on the defects identified, it might not be a problem of inspection during the Sprint Review meeting, but of the testing process. In this case, the metric *Deferred defects* can be associated with the node *Code*. Finally, it might be a false alert and ignored by the team. Therefore, this approach must be used if the team wants to use metrics just as warnings and not as direct input to the BN, which might be the case if there is not much confidence in the causality. In Table 2, we show examples of metrics that can be associated with the base BN.

3 Validation

To validate the framework in terms of completeness, we evaluated the anti-patterns presented in Eloranta et al. [4], in which an empirical study with 11 companies was executed. They identified 14 anti-patterns in adopting Scrum and their consequences, namely: (i) big requirements documentation, (ii) customer Product Owner, (iii) Product Owner without authority, (iv) long or non-existent feedback loops, (v) unordered product backlog, (vi) work estimates given to teams, (vii) hours in progress monitoring, (viii) semi-functional teams, (ix) customer caused disruption, (x) no

Table 2. Examples of metrics that can be associated with the base BN.

Node	Metrics
Code	Static analysis warnings, number of test cases, number of defects and code coverage.
Product Backlog	Percentage of ready items.
Planning meeting	Percentage of completed items.
Details	INVEST [3].
Execution	Sprint velocity.
Sprint	Running Tested Features e velocity.
Increment	Customer satisfaction, delivered value e agileEVM [18].

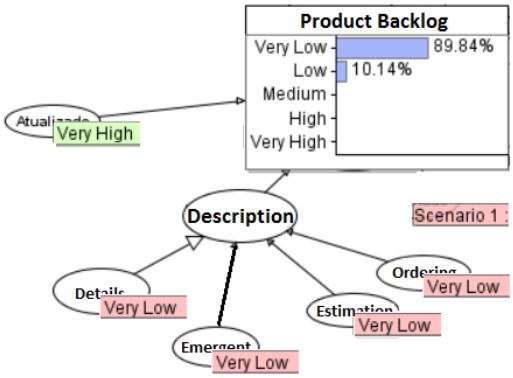


Figure 6. Results of the BN for anti-pattern (i).

sprint retrospective, (xi) invisible progress, (xii) varying sprint length, (xiii) too long sprints and (xiv) testing in next sprint. The goal of our validation was to identify if the proposed solution can identify them. Furthermore, we analyzed if the calculations are in conformance with the consequences of the anti-patterns. Due to space limitations, we only present the analysis of one anti-pattern in this paper.

For anti-pattern (i), the consequences of this pattern are that the team does not understand the requirements and that they are not ordered. It is related to the node *Product Backlog*, which is not an input node (i.e., leaf). Therefore, we analyze this pattern by defining evidences for the nodes *Details*, *Ordering*, *Emergent* and *Estimation*. As previously mentioned, in this context, the requirements are not ordered and emergent. Furthermore, if the team does not understand the requirements, it means that they are not detailed properly, as consequence, not estimated properly. By analyzing the calculated BN shown in Figure 6, we conclude that anti-pattern (i) is detected by the BN.

The anti-pattern (vi) is partially detected by the BN. The consequences of this anti-pattern are unrealistic goals and team demotivation. Even though it is related with the nodes *Estimation* and *Sprint goal*, the model does not present a causality between the quality of the estimations and the quality of the teamwork (i.e., motivation). On the other hand, the BN assists on assessing if the team is motivated in the fragment *Teamwork*. For instance, demotivated teams will probably not collaborate and share leadership.

The anti-pattern (x) is considered invalid because it is related to the Sprint Retrospective meeting, which is not modeled by the BN, as previously explained. The remaining anti-patterns are detected by the need of entering evidence on input nodes or analyzing the BN.

4 Final remarks

In this paper, we presented a framework to build a BN to assist on the assessment of Scrum-based software development methods. The BN models the main entities of the software development process and can be complemented with software practices and metrics. With the resulting BN, it is possible to monitor if the agile method is being followed and if it meets the needs of the project.

We evaluated the completeness of the solution through simulations to check if the proposed framework diagnoses 14 known Scrum anti-patterns (i.e., ScrumBut) presented in Eloranta et al. [4]. From the 14 known Scrum anti-patterns, 1 is invalid, 12 were directly detected and 1 was indirectly detected by the constructed model.

Even though our solution is an evolution of published studies [14, 13], which were evaluated in the industry, the main limitation of this study is the validation process. As future works, we will evaluate the proposed framework through case studies in the industry. To ease the usage on industry context, a tool to support the usage of the proposed framework, hiding the complexity of Bayesian networks from the practitioner, is currently under development.

The main contributions for practitioners are that the solution helps to configure and adopt Scrum-based agile methods and to develop measurement programs to continuously improve the maturity of the team and delivery process. For researchers, it can serve as a basis to configure empirical studies on Scrum-based agile methods and improve the state-of-art of agile methods measurement programs, adoption and maturity.

References

[1] S. Brinkkemper. Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, 38(4):275 – 280, 1996. Method Engineering and Meta-Modelling.

[2] A. S. Campanelli and F. S. Parreiras. Agile methods tailoring - a systematic literature review. *J. Syst. Softw.*, 110(C):85–100, Dec. 2015.

[3] S. Downey and J. Sutherland. Scrum metrics for hyperproductive teams: How they fly like fighter aircraft. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 4870–4878, Jan 2013.

[4] V.-P. Eloranta, K. Koskimies, and T. Mikkonen. Exploring scrumbutan empirical study of scrum antipatterns. *Information and Software Technology*, 74:194 – 203, 2016.

[5] N. Fenton and M. Neil. *Risk Assessment and Decision Analysis with Bayesian Networks*. CRC Press, 5 edition, 11 2012.

[6] N. E. Fenton, M. Neil, and J. G. Caballero. Using ranked nodes to model qualitative judgments in bayesian networks. *IEEE Trans. on Knowl. and Data Eng.*, 19(10):1420–1432, Oct. 2007.

[7] 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*, 97:140 – 155, 2014.

[8] R. M. Fontana, V. Meyer, S. Reinehr, and A. Malucelli. Progressive outcomes. *J. Syst. Softw.*, 102(C):88–108, Apr. 2015.

[9] S. Grapenthin, S. Poggel, M. Book, and V. Gruhn. Improving task breakdown comprehensiveness in agile projects with an interaction room. *Information and Software Technology*, 67:254 – 264, 2015.

[10] T. Javdani Gandomani and M. Ziaei Nafchi. An empirically-developed framework for agile transition and adoption. *J. Syst. Softw.*, 107(C):204–219, Sept. 2015.

[11] E. Kupiainen, M. V. Mntyl, and J. Itkonen. Using metrics in agile and lean software development a systematic literature review of industrial studies. *Information and Software Technology*, 62:143 – 163, 2015.

[12] M. Perkusich, H. O. de Almeida, and A. Perkusich. A model to detect problems on scrum-based software development projects. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 1037–1042, New York, NY, USA, 2013. ACM.

[13] M. Perkusich, K. Gorgonio, H. Almeida, and A. Perkusich. Assisting the continuous improvement of scrum projects using metrics and bayesian networks. *Journal of Software: Evolution and Process*, 2016. Article in Press.

[14] M. Perkusich, G. Soares, H. Almeida, and A. Perkusich. A procedure to detect problems of processes in software development projects using bayesian networks. *Expert Systems with Applications*, 42(1):437 – 450, 2015.

[15] R. Pichler. *Agile Product Management with Scrum: Creating Products that Customers Love*. Addison-Wesley Professional, 1 edition, 4 2010.

[16] A. Qumer and B. Henderson-Sellers. A framework to support the evaluation, adoption and improvement of agile methods in practice. *Journal of Systems and Software*, 81(11):1899 – 1919, 2008.

[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*, 58:20 – 43, 2015.

[18] T. Sulaiman, B. Barton, and T. Blackburn. Agileevm - earned value management in scrum projects. In *AGILE 2006 (AGILE'06)*, pages 10 pp.–16, July 2006.

[19] J. Sutherland and K. Schwaber. The scrum guide. http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf, 2017. Accessed in: 03-10-2017.

[20] K. Verbert, R. Babuka, and B. D. Schutter. Bayesian and dempstershafer reasoning for knowledge-based fault diagnosisa comparative study. *Engineering Applications of Artificial Intelligence*, 60:136 – 150, 2017.

[21] J. Vlietland, R. van Solingen, and H. van Vliet. Aligning codependent scrum teams to enable fast business value delivery: A governance framework and set of intervention actions. *Journal of Systems and Software*, 113:418 – 429, 2016.

[22] S. Wagner, A. Goeb, L. Heinemann, M. Kls, C. Lampasona, K. Lochmann, A. Mayr, R. Plsch, A. Seidl, J. Streit, and A. Trendowicz. Operationalised product quality models and assessment: The quamoco approach. *Information and Software Technology*, 62:101 – 123, 2015.

[23] L. Williams. What agile teams think of agile principles. *Commun. ACM*, 55(4):71–76, Apr. 2012.

[24] L. Williams, K. Rubin, and M. Cohn. Driving process improvement via comparative agility assessment. In *Proceedings of the 2010 Agile Conference*, AGILE '10, pages 3–10, Washington, DC, USA, 2010. IEEE Computer Society.

Ordering the Product Backlog in Agile Software Development Projects: A Systematic Literature Review

Ana Silva[§], André Silva[§], Thalles Araújo[§], Renan Willamy[¶], Felipe Ramos[¶], Alexandre Costa[¶], Mirko Perkusich[¶], Ednaldo Dilorenzo[§]
[§]Federal Institute of Paraíba
PB-264
Monteiro, Paraíba, Brazil, 58500-000
{anasilva.ifpb, tr.andreluis, thalleshenrique.na}@gmail.com, ednaldo.dilorenzo@ifpb.edu.br
[¶]Federal University of Campina Grande
Campina Grande, Paraíba, Brazil, 58429-140
{renanwillamy, feliperamos, antonioalexandre, mirko.perkusich}@copin.ufcg.edu.br

***Abstract*—Recently, agile software development methods have attracted the attention of academic and industrial domains. Unlike traditional approaches, agile methods focus on rapid delivery of business value to customers through fast delivery of working code. Therefore, requirement prioritization is considered a crucial process in this context, since there is the need to identify which requirements must be developed first, based on stakeholders preferences and taking into account business and technical challenges such as dependencies of tasks, developers skills, limitation of budget, etc. In Scrum, the most popular agile method, requirement management is done by maintaining the product backlog, which contains the list of functionalities that must be developed. Although some works conducted systematic reviews on the topic of agile requirement engineering, to the best of our knowledge, there is no study that reviewed the specific subject of ordering the product backlog in agile projects. Therefore, in this work, we conducted a Systematic Literature Review of studies on this topic published up to (and including) 2016 through a database search, to identify and analyze factors and techniques used to accomplish the task of ordering the product backlog in agile projects. In total, we evaluated 1556 papers, of which 13 reported on the subject of study.**

***Keywords*—Systematic Literature Review; Product Backlog; Agile Software Development; Requirements Management; Requirements Prioritization.**

I. Introduction

Due to the need of flexibility in software projects, agile software development (ASD) methods (e.g., Scrum and Extreme Programming) are becoming more popular in the last years [3]. ASD promises some benefits compared to traditional software development methods, including the delivery of business value in short iterations, following an incremental and empirical development process [10].

Moreover, ASD methods promotes the constant communication (e.g., face to face communication) and a development process open to changes [7]. Therefore, requirements are initially defined with customers, but are continuously refined [7]. However, as well as in traditional methods, a relevant activity of ASD methods is requirement engineering (RE) [10].

In the context of Scrum, the most popular agile method, requirement management is accomplished by maintaining the product backlog [7]. Therefore, to ensure a rapid delivery of business value to costumer, it is necessary to identify which requirements in this list must be developed first (ordering of requirements), i.e., to order the product backlog taking into account business and technical factors [6].

The ordering of requirements is treated as a complex multi-criteria decision making process [1], since it aims to aid the early implementation of core requirements based on the preferences of relevant stakeholders, but considering the challenges associated with software development such as limitation of budget and resources, technical knowledge of the software team, etc. Thus, it is considered to be a hard task to define and maintain [6].

Additionally, in the context of agile, requirement ordering is a continuous task, since requirements are constantly changing during iterations [10]. However, requirement engineering in agile is still informal and based on the knowledge of individuals. For example, in Scrum, the Product Owner (PO) is responsible for the elicitation and the ordering of the product backlog list [3]. Therefore, these issues can be mitigated by applying techniques to assist individuals involved in the decision-making process of ordering the product backlog.

Thus, the ordering of requirements is a crucial process in ASD projects, since it is one of the main processes to produce value quickly [6]. However, although some previous works mapped the subject of agile requirement engineering [3], [10], [7], to the best of our knowledge, research on the ordering of product backlog in ASD projects has not been systematically reviewed yet. Therefore, the main goal of our work is to conduct a Systematic Literature Review (SLR) to identify and analyze factors and techniques used to order the product backlog in ASD projects. To accomplish this work, we followed the guidelines proposed by Kitchenham and Charters [8].

DOI reference number: 10.18293/SEKE2017-007

This paper is structured as follows. In section 2, we discuss previous literature reviews in the subject of study. In Section 3, we present the protocol of our systematic review process. In Section 4, we present our findings. In Section 5, we discuss the results. In Section 6, we present our conclusions and future work.

II. Related Work

Some literature reviews were conducted on the topic of agile requirement engineering [7], [10], [6], [3]. We summarize them as follows.

Inayat et al. [7] conducted a systematic literature review focusing on practices and challenges of requirements engineering in the context of ASD. The review identified 17 practices of RE in agile, five challenges found in traditional methods which were overcame by agile requirements engineering, and eight challenges resulting from the practice of RE in agile. The authors suggest that the subject needs further investigation and more empirical results to improve its understanding.

Schön et al. [10] conducted an SLR on the topic of agile RE with a focus on the participation of stakeholders and users in the process of RE. The authors investigated existing approaches to involve stakeholders in this stage and which approaches are used to present the user's perspective during these process. They concluded that agile RE is a complex research field with cross-functional influences and that their study brings an overview of requirements management in ASD.

Heikkilä et. al [6] conducted a mapping study on the research literature of RE in ASD with focus on the benefits of this process in agile and the reported problems and corresponding solutions. The authors concluded that the agile RE definition is still vague. Among the main benefits, stand out: better requirements understanding, responsiveness to change and rapid delivery of value. Among the reported problems, stand out: user story requirements format, prioritization of requirements and imprecise effort estimation. Finally, the authors concluded that, in general, studies' evaluations are weak (i.e., need more effort in empirical evaluation) and the subject of study needs more research.

Unlike previously mentioned works, which cover the primary issues of agile RE, we focus on the identification and analysis of factors and techniques applied in the ordering of product backlogs, which is a more specific issue of agile RE.

In the area of software requirements prioritization, Achimugu et al. [1] conducted an SLR to identify and analyze existing prioritization techniques, their limitations, processes, and taxonomies. The authors concluded that existing prioritization techniques present limitations and their applicability in complex and real setting has not been reported yet. Moreover, it was considered that existing techniques need improvements. Although the work is related to ours, the authors did not focus on agile, and hence, few works were returned on this topic, which has been attracting the attention of an increasing amount of research studies and it is gaining popularity in the industry [6]. Furthermore, the authors did not evaluate the main factors used by the techniques in the ordering process, as reported in our work.

III. Review Method

In this research, we performed a Systematic Literature Review following the method defined in [8]. An SLR aims to identify, critically evaluate and integrate relevant, high-quality studies addressing one or more research questions.

A. Research Questions

As previously mentioned, the main goal of this research is to identify and analyze factors and techniques used in the ordering of product backlogs in ASD projects presented in the literature. Therefore, we formulated the following research questions (RQs):

RQ1: Which factors are considered to order the product backlog items?

RQ2: How are the factors identified in RQ1 measured?

RQ3: Which techniques are used to order the product backlog?

RQ4: Which evidences show that the techniques identified in RQ3 are efficient?

B. Data Sources and Search String

We conducted a database search in the following sources: ACM, Engineering Village, ISI Web of Science, ScienceDirect, SpringerLink, Scopus and Wiley. These data sources were chosen based on its relevance in the software engineering domain. Aiming to start to answer the research questions defined in section III-A, we formulated the following search string:

> *(prioritization OR prioritisation OR prioritizing OR prioritising OR prioritize OR prioritise OR prioritized OR prioritised OR priority OR order OR ordered OR ordering) AND (requirement OR functionality OR requisite OR prerequisite OR user story OR "user stories" OR backlog OR issue) AND (practice OR practices OR techniques OR technique OR process OR processes OR tactic OR tactics OR method OR methods OR strategy OR strategies OR factor OR factors OR component OR components) AND (software AND (agile OR gil) AND (scrum OR xp OR (crystal AND (clear OR orange OR red OR blue)) OR dsdm OR fdd OR "feature driven development" OR (lean AND (development OR desenvolvimento)) OR Kanban OR "extreme programming" OR "programao extrema" OR devops))*

The search results are presented in Table I.

TABLE I: Number of papers returned from database search.

Data Source	Results
ACM (http://dl.acm.org/)	22
Engineering Village (www.engineeringvillage.com)	43
ISI Web of Science	35
ScienceDirect (www.sciencedirect.com)	1,173
SpringerLink (www.scopus.com)	-
Scopus (www.scopus.com)	86
Wiley (onlinelibrary.wiley.com)	455
Total	1,814

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

Fig. 1: Categories of agreement or disagreement.

C. Selection Criteria

Formulating a consistent search string does not guarantee a significant set of studies. Due to database query limitations, major part of the returned papers are not related to the research subject. To select only relevant studies, we have divided the selection criteria into three phases: generic exclusion criteria, basic criteria and advanced criteria.

Generic exclusion criteria. To eliminate the most irrelevant papers we decided to discard i) papers not published in English or Portuguese languages; and ii) published in non-peer reviewed publication channel such as books, thesis or dissertations, tutorials, keynotes, etc.; and iii) duplicated.

Basic Criteria. This phase was performed by two reviewers, randomly chosen, and it consisted of reading the title and abstract of the remaining papers from previous phase (generic exclusion). Each paper was evaluated and classified following the procedure presented in Ali et al. [2], as:

- *Relevant:* papers related to agile software development and to product backlog management;
- *Irrelevant:* papers not related to agile software development or not related to product backlog management;
- *Uncertain:* the available information on the title and abstract was inconclusive or insufficient to classify as relevant or irrelevant.

According to Ali et al [2], there are six categories of agreement or disagreement between the reviewers, as shown in Figure 1.

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. Afterwards, in the next selection criteria step all doubts about the paper's relevance are clarified with a further evaluation.

Category C means that no concrete decision was made by any of the reviewers and further investigation is needed. In this case, both reviewers, independently, conduct an adaptive reading, which is composed of three steps: 1) read the Introduction; 2) if not having agreement in 1, read conclusion; 3) if not having agreement in 2, use the keywords to evaluate their usage to describe the context of the paper. This practice helps to make a decision.

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.

Advanced criteria. In this selection phase, a further investigation is performed, once a full-paper evaluation is required. We used the same criteria established in basic criteria, but reading the full-text. Each paper was evaluated by two reviewers, a data extractor and a data checker. The data extractor performs a Quality assessment (Section III-D) and a Data extraction (Section III-E). After the data is extracted, the data checker reviews to confirm if it is corrected, following the advice presented in Brereton et al. [4] and Staples and Niazi [12].

D. Quality assessment

In this stage, each paper was evaluated by a data extractor and a data checker, in the following manner: the data extractor fills the data extraction form, and then, the data checker confirmed that the data on the extraction form were correct. Each category was evaluated on a boolean scale (i.e., 0 or 1). The assessment checklist follows 11 criteria presented by Dyba and Dygsoyr [5].

E. Data extraction

During this stage, we used a spreadsheet editor to record information. For each paper, we extracted general information such as title, year and publication channel, and data related to the RQs. The following data were extracted from the papers:

(i) type of article (journal, conference),

(ii) name of the publication channel,

(iii) year of publication,

(iv) agile method,

(v) product domain,

(vi) application domain,

(vii) team size,

(viii) team distribution,

(ix) research type (based on the classification presented by Wieringa et al. [15]: validation research, evaluation research, solution proposal, philosophical papers, opinion papers or experience papers.),

(x) research question type (based on the classification presented by Shaw [11]: method or means of development; method for analysis or evaluation; design, evaluation, or analysis of a particular instance; generalization or characterization; or feasibility study or exploration.),

(xi) empirical research type (based on the classification presented by Tonella et al. [13]: experiment, observational study, experience report, case study or systematic review.),

(xii) research validation (based on the classification presented by Shaw [11]: analysis, evaluation, experience, example, persuasion or blatant assertion.),

(xiii) factors (RQ1),

(xiv) how factors were measured or identified (RQ2),

(xv) techniques (RQ3),

(xvi) evidences of techniques efficiency (RQ4).

IV. Results

In this section, we present the results for the SLR process and for the research questions as well. In Table II, we present the list of included studies. In Figure 2, we present the amount of papers per year. In Figure 3, we show the distribution of papers per type of publication channel. In Figure 4, we show the percentage of the included papers per type of agile method studied. In Figure 5, we show the aggregated results of the quality assessment.

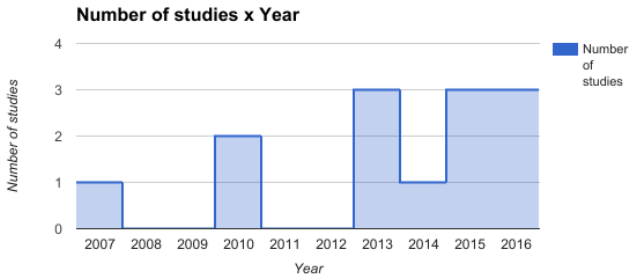


Fig. 2: Number of studies x year.

In Figure 6, we present the number of factors identified in the review (**RQ1**). We identified eight factors in our study, i.e., business value (P1, P2, P4, P5, P6, P7, P8, P9, P10, P12, P13), effort/cost (P1, P2, P4, P5, P8, P9, P13), dependency (P1, P2, P3, P4, P6, P9, P11, P13), risk (P1, P2, P9), volatility (P1, P2), technical debt (P2), human resources (P1), and schedule (P1). Additionally, we show the information related to the measurement of the factors in Table III (**RQ2**). By analyzing Table III, it is possible to conclude that effort, dependency, business value, and risk had measurement procedures presented in the works. On the other hand, the measure of four factors – volatility (P1 and P2), technical debt (P4), human resources (P1), and schedule (P1) – were not explained in the papers.

In table IV, we show the techniques to order the backlog identified in our review (**RQ3**) and the evidences that show

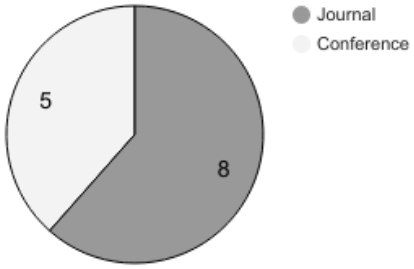


Fig. 3: Number of studies per type of publication channel.

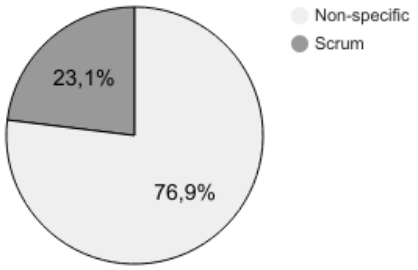


Fig. 4: Percentage of agile methods investigated in the studies.

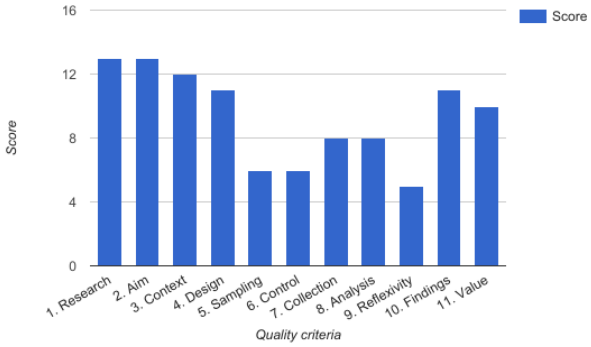


Fig. 5: Score x quality criteria.

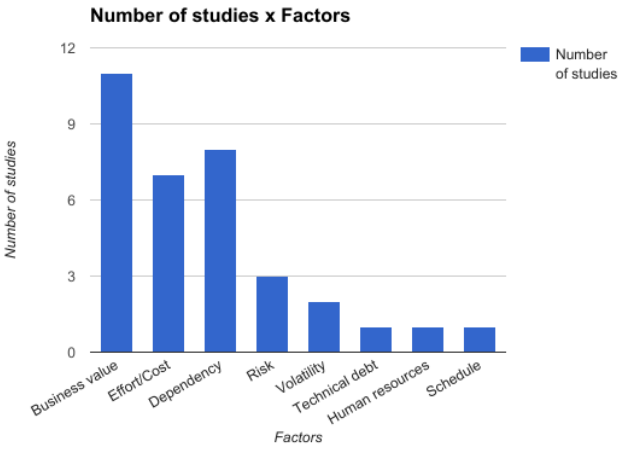


Fig. 6: Number of studies x factors.

TABLE II: Overview of the selected studies.

Paper Number	Authors	Year	Title	Publication Channel
P1	Rami Hasan AL-Taani and Rozilawati Razali	2013	Prioritizing Requirements in Agile Development: A Conceptual Framework	International Conference on Electrical Engineering and Informatics
P2	Maya Daneva, Egbert van der Veen, Chintan Amrit, Smita Ghaisas, Klaas Sikkel, Ramesh Kumar, Nirav Ajmeri, Uday Ramteerthkar and Roel Wieringa	2013	Agile requirements prioritization in large-scale outsourced system projects: An empirical study	Journal of Systems and Software
P3	Marina Trkman, Jan Mendling and Marjan Krisper	2016	Using business process models to better understand the dependencies among user stories	Information and Software Technology
P4	Michel dos Santos Soares, Jos Vrancken and Alexander Verbraeck	2010	User requirements modeling and analysis of software-intensive systems	Journal of Systems and Software
P5	Jos M. Chaves-Gonzlez, Miguel A. Prez-Toledano and Amparo Navasa	2015	Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm	Knowledge-Based Systems
P6	David P. Harvie and Arvin Agah	2017	Targeted Scrum: Applying Mission Command to Agile Software Development	IEEE Transactions on Sofware Engineering
P7	Julian M. Bass	2015	How product owner teams scale agile methods to large distributed enterprises	Empirical Software Engineering
P8	Thomas Michael Fehlmann and Eberhard Kranich	2014	Early software project estimation the six sigma way	XP 2014
P9	Weam M. Farid and Frank J. Mitropoulos	2013	NORPLAN: Non-functional requirements planning for agile processes	IEEE SoutheastCon
P10	Helena Holmstrm Olsson and Jan Bosch	2015	Towards continuous validation of customer value	XP 2015
P11	Arturo Gomez, Gema Rueda and Pedro P. Alarcn	2010	A systematic and lightweight method to identify dependencies between user stories	XP 2010
P12	Balasubramaniam Ramesh, Lan Cao and Richard Baskerville	2007	Agile requirements engineering practices and challenges: an empirical study	Information Systems Journal
P13	Jan Vlietland, Rini van Solingen and Hans van Vliet	2016	Aligning codependent Scrum teams to enable fast business value delivery: A governance framework and set of intervention actions	Journal of Systems and Software

TABLE III: Data related to the factors measurement.

Factor	Measure	Study
Effort	Person hours to complete a delivery story. Vendor uses empirical data about the productivity of their teams, if feasible.	P2
	Story points.	P8
	COSMIC function points.	P8
	PERT-based User Story Points scheme.	P9
Dependency	The dependence between user stories is identified by associating user stories with business process model activities through a method called BuPUS (Business Process User Story method).	P3
	Precedence, combination, exclusion or modification.	P5
	Dependency on key and dependency on service.	P11
Business value	A given priority level and a weight assigned to the client.	P5
	Contribution of a story card to one of the business drives on a weighted scale.	P8
	Quantitative and qualitative customer feedback techniques.	P10
Risk	Project management and requirements quality metrics, estimate and dependency risk are used to calculate the risk score.	P9

the used techniques are efficient (**RQ4**). By analyzing Table IV, it is possible to identify that only nine studies (P3, P4 P5, P6, P8, P9, P10, P11, and P13) specified the technique used to order the product backlog, the remaining three works did not give details about the used technique (P1, P2, and P12). Additionally, two studies did not show the efficiency of the used technique (P10 and P11).

V. Discussion

In this section, we discuss the results about the four research questions (see Section III-A), which explore the ordering of product backlog in ASD projects.

The first paper was published in 2008, i.e., research on this area is relatively recent. By analyzing Figure 5, we concluded that most of the papers present important findings and research value for the area, but the average quality of studies is low, considering sampling, selection of control groups, and reflexivity, which indicates that works on this topic need more empirical rigor.

We identified 13 relevant papers (see Table II), and eight different factors (see Figure 6), as follows:

Effort/Cost. P2 defines the effort based on the number of hours a person need to complete a delivery store. P8 uses the story points and the COSMIC function points to measure the effort. P9 calculates the effort based on a PERT user story points scheme. P1, P2, P5, and P13 do not give details on how the effort was measured in their work.

Dependency. P3 defines dependency between user stories by associating user stories with business process model activities through a method called BuPUS. On the other hand, P5 did not measure the dependency, but classifies it as: precedence, combination, exclusion or modification., in which the type of classification influences the resulting ordering. P11 classified the dependency into two types: dependency on key and dependency on service. P1, P2, P3, P6, P9, P11 and P13 do not give details on how the dependency was measured in their work.

Business value. P5 defines business value based on a given priority level and a weight assigned to the client. In P8, the business value is measured considering the contribution of

TABLE IV: Data related to the techniques.

Technique (RQ3)	Evidences of efficiency (RQ4)	Study
The execution orders are managed by flow and gateways elements.	According to an experiment performed with 127 undergraduate students, their solution increase the execution order and integration dependencies among user stories by the giving context.	P3
Use SysML diagrams and constructions which helps in better organizing requirements to assist on the prioritization of requirements.	They apply the solution to a list of requirements extracted from the literature and evaluate it given the IEEE Recommended Practices for Software Requirements Specification, showing better representation.	P4
Multi-objective search-based approach based on the artificial bee colony (ABC) algorithm	The authors present the results using different quality indicators and compare them with the results of other approaches published in the literature. They argue that their proposed solution is the algorithm which gives the best results regarding this multi-objective indicator for every problem instance (8 instances compared).	P5
Modifications to agile software development based on inspirations from mission command	The proposed solution was tested during a semester-long agile software engineering course designed for both graduate and upper level undergraduate students in Computer Science and Computer Engineering at the University of Kansas. They concluded that it improved planning and prioritization of requirements and developing the overall project architecture.	P6
Buglione-Trudel matrix tool	The solution was applied to a sample set of requirements.	P8
Risk-driven algorithm to prioritize and plan an improved requirements implementation sequence	The solution was applied to the requirements of the European Union procurement system just to show the different results that the algorithms calculated.	P9
Qualitative/quantitative Customer-driven Development (QCD) validation cycle	Unspecified	P10
Dependencies Identification Method	Unspecified	P11
Product Owner Group (POG), which discusses and decides about the priority of each feature on the feature backlog.The POG is headed by the Epic Product Owner.	The intervention actions (IA) resulted in an average delivery time reduction from 29 days to 10 days.	P13

a story card to one of the business drives on a weighted scale. On the other hand, P10 defines business value based on quantitative and qualitative customer feedback techniques. P1, P2, P4, P6, P7, P9, P12 and P13 do not give details on how the business value was measured in their work.

Risk. in P9, risk is defined as score based on project management and requirements quality metrics, estimate and dependency risk. P2 and P9 did not give details on how the risk was measured in their work.

Volatility. P1 and P2 uses the requirements volatility as a factor to order the product backlog, but they do not give details on how it was measured in their work.

Technical debt. In P2, technical debt is used as a factor to order the product backlog, but there is no detail about its measurement. Although the authors report that it implies the amount of architecture-redesign related work that accumulates over a period of time.

Human resources. P1 uses the human resource as a factor to order the product backlog, but there is no detail about its measurement.

Therefore, business value (11 studies), dependency (8 studies) and effort (7 studies) are the most reported factors, which shows the relevance of considering this factors in the product backlog ordering. Additionally, in the literature, Usman et al. [14] conducted an SLR on the state of the practice on effort estimation in ASD. Furthermore, Racheva et al. [9] conducted an SLR on how business value is created by agile projects.

Moreover, 9 studies give details about the techniques used in the product backlog ordering (see Table IV). None of them used the same approach. In P3, the execution orders are managed by flow and gateways elements. The authors argue that their method improve the understanding about the execution order and integration dependencies among user stories, based on an experiment performed with 127 undergraduate students. In P4, they used a SysML Requirements diagram, which details requirements relationships, and hence, aids in requirements prioritization. They apply the solution to a list of requirements extracted from the literature and evaluate it, given the IEEE Recommended Practices for Software Requirements Specification. P5 uses a multi-objective search-based approach based on the artificial bee colony (ABC) algorithm. The authors compare their approach to other ones published in literature and argue that their proposed solution reached the best results, considering 8 instances compared.

P6 uses Product Backlog grooming dialogues, which are formalized through Lines of Effort (LOEs) as visual representations of the software client's priorities and desire end state. The proposed solution was tested during a semester-long agile software engineering course designed for both graduate and upper level undergraduate students at the University of Kansas. They concluded that it improved planning and prioritization of requirements and developing the overall project architecture. In P8, the technique used was the Buglione-Trudel matrix tool, which provides agile teams with immediate feedback whether their priorities meet customer needs. The solution was applied to a set of requirements. P9 uses a risk-driven algorithm to prioritize and plan an improved requirement implementation sequence. The approach was validated through visual simulation and a case study, which indicates that the use of the proposed solution results in a decrease on the overall duration of the implementation.

P10 uses quantitative and qualitative customer feedback techniques to accomplish the task of product backlog ordering. In this study, no evidences of the technique efficiency are shown. In P11, a dependency identification method is used. In this study, no evidences of the technique efficiency are shown. P13 presents a governance framework and set of intervention actions. One of the intervention action is the Product Owner Group (POG), which discusses and decides about the priority of each feature on the feature backlog. As a result of the interventions, the average delivery time was reducted from 29 days to 10 days.

Therefore, we can conclude that there are many approaches to order the product backlog, but there is no consensus about which one achieves the best results. Additionally, in some works the efficiency of the techniques was not evaluated. Yet,

in most cases the evaluations were carried out in academic environment, which may not faithfully represent real industry environment.

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 formulated a comprehensive string, which covered keywords and their synonyms and applied it to different database searches to cover main Software Engineering conferences and journals. Another threat faced regards to researchers' opinion, which can influence in the results of the study. To mitigate this problem, each paper was evaluated by two reviewers, which should agree on the inclusion or exclusion of the paper.

VII. Conclusion and Future Work

In this paper, we conducted a systematic literature review with focus on the ordering of product backlog in agile software development projects to identify and analyze the main factors and techniques used in this process. To reach this goal, we applied a database search approach to identify the most relevant studies in the topic of study. The primary search fetched 1814 results, among them, 258 duplicated, which resulted in 1556 papers. After analyzing basic criteria, 35 papers were selected, and finally, after advanced criteria analysis (i.e., quality assessment and data extraction accomplished), only 13 papers were included in the study. At the end of the process, data from these papers were analyzed to answer research questions formulated.

We identified 8 different factors. Among them, business value (11 studies), dependency (8 studies) and effort (7 studies) were the most reported ones, which indicates that they are valuable factors to be considered in the product backlog ordering process. Additionally, we concluded there is no consensus about which technique to be applied.

Results of this study can be used as baseline by practitioners to start new works, since it presents an overview of the studies in the literature about the ordering of backlog in ASD. Furthermore, based on our results, we concluded this topic needs further investigation with the conduction of empirical studies to assess the results of applying this practice in real environments.

For future work, we intend to complement the database search with forward and backward snowballing approaches [16], by using the 13 included papers in this study as the seed set of the snowballing.

Acknowledgment

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

References

[1] P. Achimugu, A. Selamat, R. Ibrahim, and M. N. Mahrin. A systematic literature review of software requirements prioritization research. *Information and Software Technology*, 56(6):568 – 585, 2014.

[2] 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*, 97:65 – 85, 2014.

[3] W. Alsaqaf, M. Daneva, and R. Wieringa. *Quality Requirements in Large-Scale Distributed Agile Projects – A Systematic Literature Review*, pages 219–234. Springer International Publishing, Cham, 2017.

[4] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80(4):571 – 583, 2007. Software Performance5th International Workshop on Software and Performance.

[5] T. Dyb and T. Dingsyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(910):833 – 859, 2008.

[6] V. T. Heikkil, D. Damian, C. Lassenius, and M. Paasivaara. A mapping study on requirements engineering in agile software development. In *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, pages 199–207, Aug 2015.

[7] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband. A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, 51, Part B:915 – 929, 2015. Computing for Human Learning, Behaviour and Collaboration in the Social and Mobile Networks Era.

[8] B. Kitchenham and S. Charters. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE-2007-01, Schoolf of Computer Science and Mathematics, Keele University, 2007.

[9] Z. Racheva, M. Daneva, and K. Sikkel. *Value Creation by Agile Projects: Methodology or Mystery?*, pages 141–155. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[10] E.-M. Schon, J. Thomaschewski, and M. J. Escalona. Agile requirements engineering: A systematic literature review. *Computer Standards & Interfaces*, 49:79 – 91, 2017.

[11] M. Shaw. Writing good software engineering research papers: Minitutorial. In *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03, pages 726–736, Washington, DC, USA, 2003. IEEE Computer Society.

[12] M. Staples and M. Niazi. Experiences using systematic review guidelines. *Journal of Systems and Software*, 80(9):1425 – 1437, 2007. Evaluation and Assessment in Software EngineeringEASE06.

[13] P. Tonella, M. Torchiano, B. Du Bois, and T. Systä. Empirical studies in reverse engineering: state of the art and future trends. *Empirical Software Engineering*, 12(5):551–571, 2007.

[14] 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*, EASE '15, pages 12:1–12:10, New York, NY, USA, 2015. ACM.

[15] R. Wieringa, N. Maiden, N. Mead, and C. Rolland. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. *Requir. Eng.*, 11(1):102–107, Dec. 2005.

[16] 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*, EASE '14, pages 38:1–38:10, New York, NY, USA, 2014. ACM.

BPL-Framework 2.0: Support tool for Creation and Instantiation of Business Process Lines

Delacyr Almeida Monteiro Ferreira [2] Débora Maria Barroso Paiva [1*] Maria Istela Cagnin [1]

[1] College of Computing (Facom), Federal University of Mato Grosso do Sul (UFMS), Campo Grande, MS, Brazil

[2] Faculty of Exact Sciences and Technology (FACET), Federal University of Grande Dourados (UFGD), Dourados, MS, Brazil

Abstract

The use of software reuse techniques in the context of business processes, such as Business Process Lines (BPL), makes viable the efficient reuse of business process models. In this context, it is highlighted the importance of computational support to aid the creation, instantiation and evolution of BPLs, once that the models created are complex due to the domain inherent characteristics and the dynamism of the business. The objective of this paper is to present an evolution to the BPL-Framework 1.0, tool responsible for creating BPLs, to also include the instantiation of this kind of line. The tool facilitates the resolution of variability throughout the instantiation and generates the instance automatically without human intervention.

1 Introduction

The modeling of business processes aids in the comprehension and optimization of existing business processes, and also in the conception of new business processes to make organizations more competitive and efficient [1]. The resultant models from the business process modeling also provide help in the Engineering Requirements [2] once that they help in the comprehension of the business, and they aid the selection of functional and non-functional requirements for the software system adequate to attend the organization.

Even though the modeling of business processes is important, many organizations opt not to adopt this practice due to time and costs to execute it. The reutilization of business process models makes it possible to reduce the time and effort put in the elaboration of this kind of artifact, in addition to improve its quality, once they were already tested [3]. Under this perspective, the usage of software reuse techniques in the context of business processes is a mechanism to make viable the efficient reuse of business process models, such as Business Process Lines (BPL).

The BPL represents a set of closely related business processes with variabilities in their characteristics and operational contexts, instead of a single business process [4].

Basically, a BPL is composed by the following artifacts [5][6]: DOPs (business process models of the domain); a variability model where all the variabilities of all business processes that compose the BPL are represented; a Business Process Model *Template* (BPMT), which represents the flow of the processes, as well as the location of the variation points, which represent the subject of the variability [7] (for example, the color of the cars from a car maker) and its variants which represent the object of the variability [7] (for example, silver, black and white are the color options for the cars of the car maker); and finally a mapping between the variability model and the BPMT, representing the traceability between them.

The computational support for the creation, instantiation and evolution of BPLs is primordial once that the created models are complex due to the characteristics inherent to the domain and dynamism of the business. In this context, Terenciani et al. [6] developed a tool, named BPL-Framework 1.0, which is an Eclipse plugin that supports the creation and documentation of BPLs, in accordance to the approach of Management (creation, instantiation and evolution) of Business Process Lines (MBPL) [8]. This tool uses the notations of feature model [9], BPMN [10] and BPMN* [11], and manages the variability of the BPMN basic elements (activity, task and data object). Except that the BPL-Framework 1.0 does not support the instantiation of BPLs.

Two other tools were found to support the creation and instantiation of BPLs, whose instances are business process models. In one of them [12], the resolution of variabilities, during the instantiation of the line, is based on a questionnaire and the other [13] is done by the selection of graphic elements that represent the variabilities. However, both tools do not generate in a completely automatic way an instance of the line and also do not represent variability in elements of the data object type and do not offer vast validation in relation to the correctness of the artifacts of the line.

In face of the above mentioned lacks, it was noted the opportunity to evolve the BPL-Framework 1.0 to also include the instantiation of BPL, since it has compliance with an approach of management for this kind of line and already supports the creation of BPLs based on consolidated notations both for representation of variabilities and representation of business process models. This way, the objective of this work is to present the BPL-Framework 2.0, which is an evolution to the BPL-Framework 1.0, that also include the instantiation of BPLs. The results of a performed evaluation

*Financial support by Fundect (T.O. nº 219/2014 and 102/2016)

DOI reference number: 10.18293/SEKE2017-080

showed that the tool satisfactorily meets the quality requirements of a software product from the ISO/IEC 25010 [14]. The tool also offers support to the resolution of variability of the BPL by selection, generates automatically an instance without the need of human intervention and validates the artifacts of the line.

2 BPL-Framework 1.0

BPL-Framework 1.0 is a computational tool to support the creation and documentation of BPLs. It supports the documentation of the BPL, since it allows the elaboration of its artifacts, by means of a computational tool, such as DOPs in the BPMN notation [10], variability model in the feature model notation [9], and BPMTs in the BPMN* notation [6]. The BPMN* is an extension to the BPMN notation to represent variabilities on activities, tasks and data objects of business process models; and it also gives control over traceability between the feature model and the BPMT.

The BPL-Framework 1.0 was developed as a open-source plug-in for the Eclipse IDE. It was composed by the BPMN2 Modeler plugin [15], for the elaboration of DOPs, and by the FeatureIDE plugin [16], for the elaboration of the feature model.

The BPMN2 Modeler plug-in is composed by the Graphiti plug-in [17], which is a graphical structure for the development of diagram editors, and by the metamodel generated by the Eclipse Modeling Framework (EMF) [18]. This metamodel is created and defined in the *ecore* format, which is basically a subset of class diagrams. From the *ecore*, it is possible to generate Java code compatible with the specifications from the BPMN 2.0.2 notation [10].

Throughout the development of the BPL-Framework 1.0, the BPMN2 Modeler was extended to support the BPMN* notation such that the tool could aid the elaboration of the BPMT. For that, five new attributes were added to the classes of the *ecore* from the EMF to make configurations of variability: i) *IsVarpoint*: represents variation points in the BPMT; ii) *IsVariant*: represents the variants in the BPMT; iii) *VarPointType*: represents types of variation points in the BPMT (*AND, XOR* and *OR*); iv) *FeatureId*: enables traceability between the feature model and the BPMT; v) *FeatureType*: represents optional and mandatory elements in the BPMT (optional, mandatory and none), according to the features configurations used by the FeatureIDE.

3 BPL-Framework 2.0

The BPL-Framework 2.0 was developed to support both creation (already included in the version 1.0) and instantiation of BPLs. During the instantiation, the business analyst must choose an existing BPL that he/she wants to instantiate and the relevant BPMTs. The tool then generates automatically a configuration model for each selected BPMT. After that, the business analyst can start the variability resolution for each configuration model.

The business analyst must analyze each variation point in the configuration model and select the necessary variants for the instantiation of the BPL according to the business process of the organization. Subsequently, the DOPs are generated based on the selected variants from configuration models considered during the instantiation of the BPL.

There can be commonalities in the configuration model that will be part of all DOPs. The green color was used to represent commonalities and variants selected of activity and sub-process types. The orange color was used to represent commonalities and variants selected of data object type.

When resolving the variabilities by selecting the adequate variants to attend the objectives of the organization's business, it is possible to encounter some situations. When the variation point (Task 2 from Figure 1(a)) has one or more variants (Tasks 3 and 4 from Figure 1(a)), but its variability association has the $<<xor>>$ stereotype, it indicates that only one of the variants can be selected.

Another situation is observed when the stereotype of the variability association between a variation point (Task 5 from Figure 1(a)) and its variants (Tasks 6, 7 and 8 from Figure 1(a)) is from the $<<or>>$ type. When this happens, the business analyst has the possibility of selecting one or more variants from this variation point. The analyst must configure the properties for each selected variant, as shown in Figure 1(b), for the proper validation of the configuration model.

If every variant from a variation point with the $<<or>>$ has the attribute *Sequence* with distinct values, the flow of these variants in the DOP generated based on the configuration model with the solved variabilities will obey the order defined by the values of this attribute. If there is a set of selected variants that has two or more equal values from the *Sequence* attribute, it is up to the business analyst to inform the type of *Gateway* that will represent this set of variants. This data can be informed by the business analyst at any time during the instantiation.

If the chosen gateway is of the XOR type (*Exclusive*), it represents a flow where only one of the paths will be taken, according to conditions to be verified. If the chosen gateway is of the *Parallel* type, it represents a flow where two or more paths can be executed in parallel. If the chosen gateway is of the OR (*Inclusive*) type, it represents a flow where there can be a combination of the paths, based on the conditions to be verified, according to the values of the *Condition* attributes. For example, the values of the *Condition* attributes from Task 7-1 and Task 8-1 illustrated on Figure 1(b).

When the variability happens on an element of the data object type with variability associations of the $<<or>>$

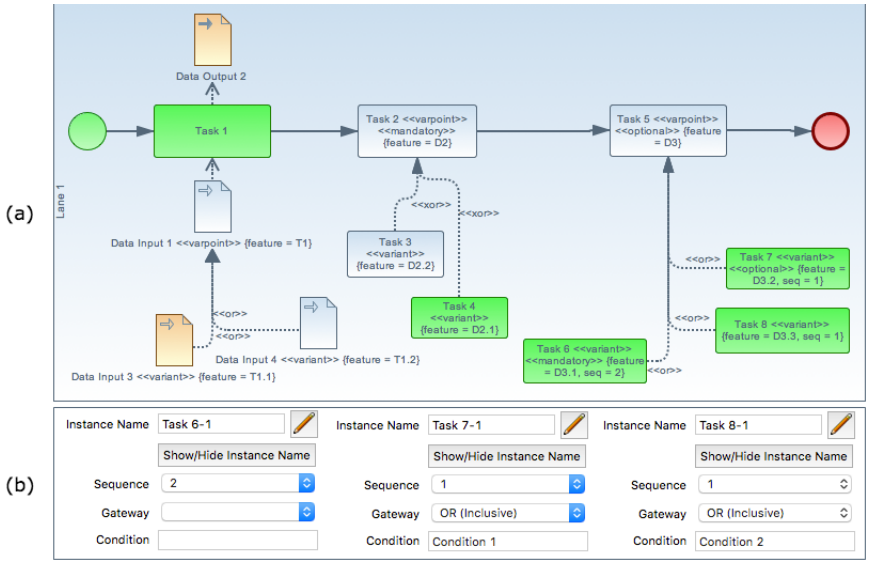


Figure 1. Configuration model with commonalities, variation points and variants.

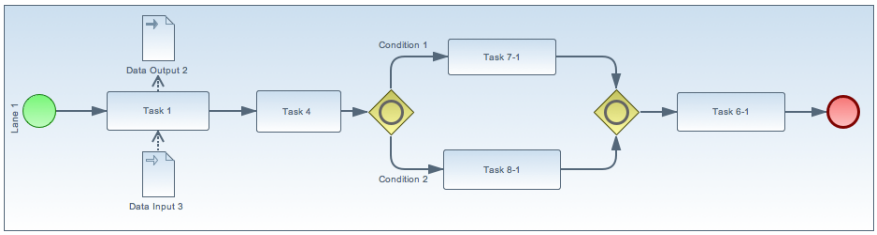


Figure 2. DOP obtained from the configuration model from Figure 1(a).

type (Data Input 3 from Figure 1(a)), there is no need to inform values for the *Sequence* attribute, once that all selected elements will be directly connected to the activity or sub-process by means of the data association during the automatic generation of the DOP.

After finishing the resolution of variabilities in the configuration models of the BPL, the business analyst may ask the tool to generate the DOP for each configuration model. Figure 2 presents the DOP generated automatically by the tool, without human intervention, based on the configuration model from Figure 1.

It is highlighted that once all variabilities were solved by the business analyst, the configuration models used for the generation of the DOPs are kept for reuse purposes on future instantiations, enabling the business analyst to take existing configuration models as a base to generate the DOP of an organization, reducing the time and costs of the instantiation of the BPL.

4 BPL-Framework 2.0 Evaluation

The objective of the evaluation conducted was to evaluate the functionalities of the BPL-Framework 2.0 through the instantiation of a BPL in the rental domain, based on: i) functional suitability sub-characteristics (accuracy), legibility and usability (ease of learning, ease of use and attractiveness) from the ISO/IEC 25010; and ii) reusability of business processes by means of BPLs instantiation supported by the BPL-Framework 2.0, by the point of view of the business analyst.

Fourteen evaluators were selected to join the evaluation of the BPL-Framework 2.0. Evaluators received a two hours training about variability, features model and essential concepts about BPL, in addition to an overview of the functionalities of the BPLs creation module of the BPL-Framework. The evaluators main task was to instantiate a BPL from the rental domain [6] based on a given description about a business process of rental of dresses. At the end of the instantiation, the evaluators answered an evaluation form, which contained a set of questions related with each defined evaluation requirement.

The data from the evaluation forms answered by the evaluators were charted and analysed. For the learnability requirement, the attribute that corresponds to the ability to understand how the tool works, 57% of the evaluators were satisfied and the other 43% were completely satisfied.

As for the attractiveness, the attribute that points to the capacity of the tool to be attractive for the user, 78,5% of the evaluators were satisfied, 14,5% were completely satisfied and only 7% of the evaluators showed to be partially satisfied. The justification from these last evaluators pointed to

the low number of informative messages during the usage of the tool. Concerning the accuracy, which evaluates the capacity of the tool to obtain, with the necessary precision degree, the DOP according to the description of the business process, 50% of the evaluators were satisfied and the other 50% were completely satisfied. The reason for this is that the tool performs automatic validation of the configuration models to guarantee that all variabilities are solved before starting the generation of the DOP.

The legibility requirement allowed to evaluate the evaluators' satisfaction towards the ability of the BPL-Framework 2.0 to allow the making of legible DOPs. Some practices were added to provide more legibility while obtaining the DOPs, such as emphasizing, through the use of colors, the selected variants in the configuration model. This way, 21,5% of the evaluators were completely satisfied and 57% of the evaluators were satisfied. On this requirement, the other 21,5% of the evaluators were partially satisfied, once they reported having had to change the relative position of some elements in the obtained DOP to improve visibility, once that some elements from model automatically generated by the tool were not completely aligned with the flow.

Another requirement in the evaluation of the BPL-Framework 2.0 was related to the reusability of the business process models by means of BPL, where 78,5% of the evaluators were satisfied, 14,5% were partially satisfied and the other 7% of the evaluators were completely satisfied. The evaluators affirmed that the tool made simpler the instantiation of a BPL, since it enables the selection of the desired variability in a graphic and visual way, easing the resolution of variability. Besides that, the tool is equipped with an algorithm that automatically adjusts, in the best possible way, the position of the elements in the generated DOP to make it the most readable as possible.

5 Conclusions and Future Work

This paper presented a computational tool to support the creation and instantiation of BPLs. The evaluation performed evaluated the BPL-Framework 2.0's instantiation module. The results from this evaluation allowed to note that the tool enables the instantiation of BPLs with adequate accuracy, legibility and usability. In particular, it enables the resolution of variabilities to be done visually by the selection of variants suitable to the organization, it validates the configuration models, generates automatically an instance of the BPL, without the need of human intervention, and provides the configuration models with solved variabilities to also be reused. In the future, we plan to implement a mechanism of version control for the artifacts of the BPLs managed by the BPL-Framework 2.0, and to extend the BPL-Framework 2.0 to allow the evolution of BPLs, in addition to the integration with software assets repositories, aiming to ease the management of this kind of line.

References

[1] K. Laudon and J. Laudon, *Management Information Systems*, 12th ed. São Paulo: Prentice Hall, 2011.

[2] R. S. Wazlawick, *Object-Oriented Analysis and Design for Information Systems*, 3rd ed. Campus, 2015, in portuguese.

[3] S. A. Z. Ladeira, R. Dellosso, R. T. V. Braga, and M. I. Cagnin, "Reuse of Bussiness Modeling based on Views: A Case Study," in *Brazilian Symposium on Software Engineering*, 2008, pp. 140–155, in portuguese.

[4] N. Boffoli, D. Caivano, D. Castelluccia, and G. Visaggio, "Business process lines and decision tables driving flexibility by selection," in *International Conference on Software Composition*. Springer, 2012, pp. 178–193.

[5] G. Gröner, M. Bošković, F. S. Parreiras, and D. Gašević, "Modeling and validation of business process families," *Information Systems*, vol. 38, no. 5, pp. 709–726, 2013.

[6] M. F. Terenciani, G. B. Landre, D. M. B. Paiva, and M. I. Cagnin, "A plug-in for eclipse towards supporting business process lines documentation," in *International Conference of Computer Systems and Applications*, Marrakech, Morocco, 2015, pp. 1–8.

[7] K. Pohl, G. Bockle, and F. van der Linden, *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer-Verlag Berlin Heidelberg, 2005.

[8] G. B. Landre, "GLPN: An Approach for Business Processes Lines Management," Master's thesis, College of Computing, Federal University of Mato Grosso do Sul, Campo Grande-MS, Brazil, 2012.

[9] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-Oriented Domain Analysis (FODA): Feasibility Study," Carnegie-Mellon University, Software Engineering Institute, Pittsburgh, USA, Tech. Rep., 1990.

[10] O. M. G. OMG, "Business Process Model and Notation (BPMN)," 2013. [Online]. Available: http://www.omg.org/spec/BPMN/2.0.2/

[11] M. F. Terenciani, D. Paiva, G. Landre, and M. I. Cagnin, "BPMN* - A Notation for Variability Representation in Business Process Towards Supporting Business Process Line Modeling," in *International Conference on Software Engineering and Knowledge Engineering*, Pittsburgh, USA, 2015, pp. 1–4.

[12] F. Gottschalk and M. L. Rosa, *Process configuration*. pp 459-487, 2010, vol. Modern Business Process Automation.

[13] R. Cognini, F. Corradini, A. Polini, and B. Re, "Process variability modeling for complex organizations," in *International Conference on Enterprise Systems*, Basel, Switzerland, 2015, pp. 9–20.

[14] ISO, "Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models," International Organization for Standardization, ISO 25010:2011, 2011.

[15] M. Gille, "BPMN2 Modeler," 2013. [Online]. Available: http://eclipse.org/bpmn2-modeler/

[16] T. Thüm, C. Kästner, F. Benduhn, J. Meinicke, G. Saake, and T. Leich, "FeatureIDE: An Extensible Framework for Feature-Oriented Software Development," *Science of Computer Programming*, vol. 79, pp. 70–85, 2014.

[17] M. Wenz, "Graphiti - a Graphical Tooling Infrastructure," 2015. [Online]. Available: https://eclipse.org/graphiti/

[18] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework*, 2nd ed. Addison-Wesley Professional, 2015. [Online]. Available: http://www.eclipse.org/modeling/emf/

Practical similarities and differences between Systematic Literature Reviews and Systematic Mappings: a tertiary study

Bianca M. Napoleão
Depart. of Computer
Federal Tech. Univ. of Paraná
Cornélio Procópio, PR – Brazil
biancanapoleao@alunos.utfpr.edu.br

Katia R. Felizardo, Érica F. de Souza
Depart. of Computer
Federal Tech. Univ. of Paraná
Cornélio Procópio, PR – Brazil
katiascannavino,ericasouza@utfpr.edu.br

Nandamudi L. Vijaykumar
Lab. of Comp. and Applied Math.
Nat. Inst. for Space Research
São José dos Campos, SP – Brazil
vijay.nl@inpe.br

***Abstract*—Background: Several researchers have reported their experiences in applying secondary studies in Software Engineering (SE), however, there is a lack of studies discussing the distinction between Systematic Mappings (SMs) and Systematic Literature Reviews (SLRs). Aims: The objective of this paper is to present the results of a tertiary study conducted to collect and evaluate evidence to better understand similarities and differences between SLRs and SMs related to four aspects: research question, search string, search strategy and quality assessment. Method: We identified 170 secondary studies that were reviewed to answer a set of Research Questions (RQ) related to the practical conduction of secondary studies in SE. Results: Results show that both SLRs and SMs have generic RQs, broad search strings, and adopt automatic search as search strategy. However, quality assessment has been more widely adopted in SLRs. Conclusions: In practice, only the quality assessment is conducted differently in SLRs and SMs.**

Keywords*—*Systematic Literature Review, Systematic Mapping, Secondary Studies

I. Introduction

Evidence-Based Software Engineering (EBSE) employs appropriate research methods to build a body of knowledge about when, how, and in what context methods or tools are more appropriate to be employed for Software Engineering (SE) practice. In 2004, EBSE was first introduced as a means to advance and improve the discipline of SE [1]. In this context, Systematic Literature Reviews (SLRs) [2] and Systematic Mappings (SMs) [3] have been providing methodological and structured processes to identify and aggregate research evidence. SLRs and SMs are said to be secondary studies [2]. They are studies that review primary studies related to some research questions (RQs) to integrate/synthesize evidence related to those RQs. A primary study is related to an individual publication or a study, such as an empirical study, which investigates a specific RQ. The advantages of performing secondary studies include, among others, reduced likelihood of bias in results and the potential ability to combine data from several studies. These benefits are some reasons why secondary studies have been gaining popularity over the years [4].

DOI reference number: 10.18293/SEKE2017-069

Kitchenham et al. [2] and Petersen and colleagues [3] recognize that currently, the distinction between SMs and conventional SLRs conduction is somewhat fuzzy. The main contribution of our paper is to establish a fair and practical discussion about the conduction of secondary studies in SE area. Several studies have reported experiences and lessons learned from researchers conducting secondary studies [3], [5], [6], [7]. However, these studies do not focus on the practical similarities and differences between secondary studies. Therefore, we performed a tertiary study in identifying practical similarities and differences between SLR and SM. The findings from this study are expected to contribute to the existing knowledge with respect to the conduction of secondary studies in SE. Some of the concrete doubts that we intend to clarify are: *1– What types of Research Questions (RQs) have been answered in SLRs and SMs studies? Are these RQs generic or specific?*; *2 – How the search strings have been formulated?*; *3 – Which search strategy has been adopted in SLRs and SMs studies?*; and *4 – Has the quality of the studies been assessed?*

The remainder of this paper is organized as follows. Section II presents a brief overview of secondary studies. Section III discusses the research method applied to perform our tertiary study. Subsections III-A and III-B discusses the results, their implications, and limitations. Finally, Section IV concludes the paper and presents directions for future work.

II. Background

Secondary studies (SLR and SM) are means of finding, critically evaluating and interpreting available research papers (primary studies) on a particular set of RQs, topic area, or a phenomenon of interest. The method is intended to ensure that the literature review is unbiased, rigorous and auditable [8].

The SLR process comprises three main phases [2]: *(i) Planning:* refers to the pre-review activities, and establishes a review protocol, besides conducting a pilot test; *(ii) Conducting:* aims at searching and selecting the studies, in order to extract and synthesize data from them. This phase comprises the following activities: study searching, study selection, study quality assessment, and data extraction; *(iii) Reporting:* is the final phase, and aims at writing up the results and circulating them to potentially interested parties.

A SM is a more open form of SLR, providing an overview of a research area to assess the quantity of evidence existing

on a topic of interest [2]. In general, SM is conducted by a planning phase, which includes formulation of RQs and definition of inclusion and exclusion criteria, followed by search and screening of primary studies. The *data extraction* activity for SM is broader than the data extraction process for SLR and the analysis of a mapping does not include the use of depth analysis techniques, such as meta-analysis, but rather summaries. In a nutshell, SM deals with a broader research topic while SLR deals directly with a specific RQ narrowing the search for specific answers.

As can be observed in Table I, SLRs and SMs are different in various aspects, such as, goals, breadth, validity issues and implications. They can also vary in more basic aspects of their breadth and depth:

– **Focus of the review:** The focus of an SLR is to aggregate primary studies in terms of its results and investigate whether these results are consistent or contradictory. SLRs are performed to synthesize evidence. On the other hand, SMs provide a broader view of a research topic and identify both clusters (group of studies related to a same theme – may be suitable to undertake an SLR on this theme) and subtopics in which more primary studies are needed to be developed.

– **Research Questions:** SLRs focus on very narrow questions, while SMs focus on broad questions, which may limit the analysis, interpretation and generalization of the findings. SLRs aggregate results related to specific research questions. SMs have an objective to find and classify primary studies in subtopics.

– **Methods for searching:** Both SLRs and SMs attempt to be exhaustive in finding all relevant studies. The aim is to be able to answer the review questions by including all relevant research, however, while SLRs generally look at one type of evidence, i.e., empirical studies, SMs may include many different types of primary research, i.e., empirical studies, technical reports, theses, among others.

– **Methods for selecting:** One of the main differences between SLR and SM is related to studies selection activity. The scope of an SM is broader and the analysis and synthesis more general than in an SLR. SMs involve more studies to be selected while SLRs involve fewer studies, but they should be analyzed in a greater depth.

– **Methods for data extraction:** For undertaking an SM the data to be collected from the primary studies could include: (i) bibliographic information on the publications in which the primary studies are reported, e.g. Journal title, publication year; and (ii) basic data to describe what research has been done and how, e.g. country of the study, technique used, among others. For an SLR the data one collects from the primary studies could include: (i) detailed data on the methods and results of each study; (ii) a structured description of each study; (iii) the results (findings) of each study.

– **Synthesis:** SMs simply describe basic details about each primary study and variables can be used in coding the studies. The description may include, e.g., methods used, geographical distribution of the studies, year of the publication. SMs tabulate primary studies into categories. SMs may synthesize all, or part of the research studies described in a map. In SLRs the main interest is a full synthesis of results.

III. Tertiary study on practical similarities and differences between SLRs and SMs

Although there are publications defining how to perform SLRs and SMs, such as [2] and [3], there is still need for studies addressing different aspects of the conduction of SLRs and SM in practice. In order to evaluate similarities and differences between SLR and SM studies we performed a tertiary study, which is presented in this section. Tertiary study is considered as a review that focuses only on secondary studies, i.e., it is a review about other secondary studies [8].

First of all, we created our search string, described as: (*"software engineering"*) AND (*"systematic literature review'* OR *"systematic review"* OR *"systematic mapping"* OR *"mapping study"* OR *"systematic literature mapping"*).

After the definition of the search string, the process of identifying relevant literature was started. We chose the databases based on criteria, such as: (i) coverage (large number of conferences proceedings and journals in different knowledge areas); (ii) content update (publications are regularly updated); (iii) availability (full text of the papers are available); (iv) quality of results (accuracy of the results obtained by the search); and (v) versatility to export (a mechanism to export the results of the search is available). The most commonly used databases in SE area that meet to the characteristics described above are: IEEE Xplore *(http://ieeexplore.ieee.org)*; ACM Digital Library *(http://dl.acm.org)*; ScienceDirect *(http://www.sciencedirect.com/)*; Scopus *(http://www.scopus.com/)*; and Compedex *(http://www.engineeringvillage2.org)*.

Studies were included in the study if they met the following inclusion criteria (IC): **IC1:** The study should be in the SE area; and **IC2:** The study should present a protocol and/or the description of its conduction.

With respect to the exclusion criteria (EC), studies were excluded if: **EC1:** The study does not have an abstract; **EC2:** The study is just published as an abstract; **EC3:** The study is not written in English; **EC4:** The study is an older version of other study already considered; **EC5:** The study is not a scientific study, such as editorials, summaries of keynotes, workshops, and tutorials; **EC6:** The study is not a secondary study; **EC7:** The study is a book chapter or a guide.

A total of 970 studies were identified during the search for evidence, including 355 duplicates. Out of these studies, we selected secondary studies by reading their titles and abstracts and applying the inclusion and exclusion criteria. As a result, a total of 490 studies were selected and 125 were excluded. Finally, the 490 papers were read in full and inclusion and exclusion criteria were again applied, resulting in 320 studies being rejected. Thus, we identified 170 relevant studies from the five sources that we searched.

With the final set of secondary studies, the data extraction and synthesis activities were carried out on all 170 papers. We planned to perform the data synthesis for our RQs using tables (totals and summaries). The RQs are answered and described in the next section.

A. Results

In order to answer the RQs we created a data extraction form using Microsoft Excel software to store relevant informa-

TABLE I. COMPARING SLR AND SM (ADAPTED FROM [2], [3]).

Features	SLR	SM
Focus of the review	– Identify, analyze and interpret all available evidence related to a specific RQ	– Identify and classify what evidence is available (broad review) in a specific topic of area
	– Identify best practices based on empirical evidence	– Establish the state of evidence
Research Questions	– Narrow RQs	– Broader RQs
	– Specific RQs – Consider population; intervention; comparison and outcomes (PICO)	– Multiple RQs – Consider only population and intervention
Methods for searching	– Search string highly focused	– Search string less highly focused
Methods for selecting	– Generally few studies are considered	– A large number of studies are considered (broad coverage)
	– The studies are evaluated in details	– The studies are not evaluated in details
Methods for data extraction	– The primary studies are assessed regarding their quality (the main goal is to establish the state of evidence)	– The primary studies are not assessed regarding their quality
	– Include data extraction procedures – It is a time-consuming task	– Much broader (classification and categorization stage) – It is not a time-consuming task
Synthesis	– Include depth analysis techniques, e.g., meta-analysis and narrative synthesis	– Include no-depth analysis techniques, e.g., total and summaries
Dissemination of the results	– Higher importance for practitioners (relevant to industry)	– May be more limited, the aim is to influence the future of the research in a specific topic

tion. The form was filled with data extracted from each study and it is available on *https://goo.gl/Uh5jck.*

The data was grouped into categories. This categorization was supported by Microsoft Excel software to filter data for analysis and results visualization. Therefore, it can be said that the adopted classification scheme emerged from the selected studies.

The following is a brief description of the results. First of all, we identified the secondary studies types. The results are showed in Table II. The majority of studies are SLRs (approximately 70%) and around 30% are SMs. We identified three updated SLRs, however they were not separately categorized, i.e., they were similar to the other studies and were categorized according to their type: SLR or SM.

TABLE II. SECONDARY STUDIES: TYPE

Study Type	Quantity	Percentage
SLR	118	69.41%
SM	52	30.59%

Moreover, we verified the publication type to know what are the most common targets in which these studies are published. The majority of the studies were published in Conferences (53.53%) followed by Journals (33.53%). More detailed findings can be observed in Table III.

TABLE III. SECONDARY STUDIES: LOCAL OF PUBLICATION

Publication Type	SLR (%)	SM (%)	Total (%)
Conference	62 (52.54%)	29 (55.77%)	91 (53.53%)
Journal	40 (33.90%)	17 (32.69%)	57 (33.53%)
Congress	2 (1.69%)	0 (0%)	2 (1.18%)
Symposium	11 (9.32%)	5 (9.61%)	16 (9.41%)
Workshop	3 (2.54%)	3 (2.54%)	4 (2.35%)

With an objective to show an overview of studies area, we followed "The Guide to the Software Engineering Body of Knowledge (SWEBOK)", which describes generally accepted knowledge about software engineering [9], to classify the studies according to SE subareas. We created 11 subareas based on SWEBOK and another one labeled as "others". In particular, this category includes studies not classified in other available areas, such as, studies about specific SE tools and experiments. The defined categories and the number/percentage of studies classified in each category can be visualized in Table IV.

As shown in Table IV the majority of SLRs and SMs are related to Software Requirements (17.65%) and Software Engineering Models and Methods (17.06%), followed by Software Construction (10.59%) and Software Engineering Professional Practice (10.00%). Considering only SLRs, the Software Requirements area (19.49%) remaining as the main area of publication. On the other hand, in SM context, the Software Engineering Models and Methods is the area of major interest (19.23%). Only a few of SLRs and SMs are on Software Maintenance, adding three SLRs and two SMs, totaling 2.94% of the inclued studies.

In the sequence, the RQs will be answered in details.

RQ1: *What types of research questions (RQs) have been answered in SLRs and SMs studies?*

We assessed the RQs of SLRs and SMs considering two main aspects: (i) RQ granularity: generality or specificities; and (ii) RQ formulation: the use of PICO (**P**opulation, **I**ntervention, **C**omparison, **O**utcome) to structure RQs.

A RQ describing the main objective of a research is a generic question. Jia and Yu [10] suggest that the 5W+1H model can be used to formulate RQs in secondary studies, consequently generic questions are initiated with W-words and H-words, such as, "Why", "Who", "What", "Why", "Where", "When", "How", "How many". For example, in the SM performed by Paz et al. [11] the following generic RQs were created: "*What are the most commonly used methods to evaluate the usability of software applications in the context of a development process?*" and "*What types of applications are frequently reported in the literature as part of a usability evaluation in software developments?*".

On the other hand, a specific RQ highlights particularities of an investigated research topic. For example, the SLR performed by Engström et al., [12] presents one generic RQ ("Which techniques for regression test selection in the literature have been evaluated empirically?") and three specific RQs: (i) "Can these techniques be classified, and if so, how?"; (ii) "Are there significant differences between these techniques that can be established using empirical evidence?"; and (iii) "Can technique A be shown to be superior to technique B, based on empirical evidence?". It can observed that the specific RQs are related to the generic one.

TABLE IV. SECONDARY STUDIES: STUDIES AREA

Study Area	SLR (%)	SM (%)	Total (%)
1- Software Requirements	23 (19.49%)	7 (13.46%)	30 (17.65%)
2- Software Design	8 (6.78%)	8 (15.38%)	16 (9.41%)
3- Software Construction	11 (9.32%)	7 (13.46%)	18 (10.59%)
4- Software Testing	4 (3.39%)	4 (7.69%)	8 (4.71%)
5- Software Maintenance	3 (2.54%)	2 (3.85%)	5 (2.94%)
6- Software Management	8 (6.78%)	2 (3.85%)	10 (5.88%)
7- Software Engineering Process	9 (7.63%)	0 (0%)	9 (5.29%)
8- Software Engineering Models and Methods	19 (16.10%)	10 (19.23%)	29 (17.06%)
9- Software Quality	5 (4.24%)	2 (3.85%)	7 (4.12%)
10- Software Engineering Professional Practice	14 (11.86%)	3 (5.77%)	17 (10.00%)
11- Software Engineering Economics	4 (3.39%)	7 (13.46%)	11 (6.47%)
12- Others	7 (5.93%)	3 (5.77%)	10 (5.88%)

Overall, the results (shown in Table V) indicate that both SLRs and SMs presented more generic RQs (SLR 83.05% and SM 86,54%) than specific RQs. Nevertheless SLRs presented a slightly larger number of specific RQs (16,95%) than SMs (13,46%).

TABLE V. SECONDARY STUDIES: GENERIC AND SPECIFIC RQS

Study Type	Generic RQs (%)	Specific RQs (%)
SLR	98 (83.05%)	20 (16.95%)
SM	45 (86.54%)	7 (13.46%)

According to [2], PICO can help the RQs structuring. In summary, our results showed that PICO is used only in 26.27% of SLRs and in 32.69% of SMs (see Table VI). We can conclude that most of the SE researchers are not using PICO to structure their RQs.

TABLE VI. SECONDARY STUDIES: USE OF PICO

Study Type	Yes (%)	No (%)
SLR	31 (26.27%)	87 (73.73%)
SM	17 (32.69%)	35 (67.31%)

RQ2: *How the search strings have been formulated?*

In order to evaluate how search strings have been formulated, we considered three aspects, described following: (i) the description of the search string in the paper; (ii) the quantity of logical operator AND used to compose the search string; and (iii) the validation of the search string through the conduction of a pilot test.

The search string is fundamental to find relevant studies about a specific area or topic [2]. Approximately in 74% of SLRs and 77% of SMs (see Table VII) the search string used was not described, not even an external link containing this information was provided. In some secondary studies, such as, [13] and [14], only generic terms used for searches were described, however the formalization of the search string containing logical operators and the list of synonyms is not shown.

TABLE VII. SECONDARY STUDIES: SEARCH STRING DESCRIPTION

Study Type	Yes (%)	No (%)
SLR	87 (73.73%)	31 (26.27%)
SM	40 (76.92%)	12 (23.08%)

During the definition of the search string, the focus is the identification of terms related to the research topic that are commonly used in primary studies. A well accepted practice for formulating the search string is to identify related terms that can be considered synonyms concatenating them using the logical operator OR. Subsequently, each group of terms is concatenated with the other groups using the logical operator AND.

As can be noted in Table VIII, 43.68% of SLRs search strings were formed by only one AND logical operator and 27.59% were formed by two ANDs. Similarly, 32.50% of SMs search strings were also formed by only one AND logical operator, and a total of 37.50% were formed by two ANDs.

The use of the logical operator AND tends to restrict the number of studies returned, since the result of the operation is TRUE only if all rows of the truth table contains values TRUE (e.g. term A=true AND term B=true THEN study retrieved). As shown in line two of Table VIII in seven SLRs their search strings were formed only by terms and their synonyms, without any AND operator, characterizing a fairly generic string.

TABLE VIII. SECONDARY STUDIES: SEARCH STRING AND LOGICAL OPERATORS

Quantity	SLR	MS
0	7 (8.05%)	1 (2.50%)
1	38 (43.68%)	13 (32.50%)
2	24 (27.59%)	15 (37.50%)
3	6 (6.90%)	5 (12.50%)
4	6 (6.90%)	4 (10.00%)
$\geq$5	6 (6.90%)	2 (5.00%)

The pilot test, according to [2], aims to validate each one of the items defined in the protocol. The test verifies the feasibility and execution of the study allowing the identification of necessary modifications. During the execution of the pilot test, it is common to note, for example, the absence of terms in the search string that may lead to the non-identification of relevant studies. Our results, presented in Table IX, revealed that the vast majority of both SLRs and SMs did not conduct the pilot test. Only 20.34% of SLRs and 11.54% of SMs did it. The pilot test was conducted more times in SLRs than SMs.

TABLE IX. SECONDARY STUDIES: SEARCH STRING PILOT TEST CONDUCTION

Study Type	Yes (%)	No (%)
SLR	24 (20.34%)	94 (79.66%)
SM	6 (11.54%)	46 (88.46%)

RQ3: *Which search strategy has been adopted in SLRs and SMs studies?*

With respect to RQ3 it is possible to note that the most employed search strategy in conducting secondary studies in SE is the automatic search (see Table X). A total of 70 of the 118 SLRs (59.32%) and 24 of the 52 SMs (46.15%) used the automatic search as exclusive search strategy.

Although the automatic search is the most adopted strategy, it presents difficulties in its use, for example, a challenge is the definition of the search string [15]. Only the use of the automatic search may not be sufficient for identification of all relevant studies. This leads researchers to explore other strategies such as complementary mechanisms to extend the identification of relevant studies. The use of automatic search combined with other strategies is adopted in 36.44% of SLRs and in 48.07% of SMs (see lines four, five and six in Table X).

Only 5 SLRs (4.24%) and 3 SMs (5.77 %) adopted exclusively manual search. The snowballing strategy, which does not use search string, was not exclusively used. Some authors argue that snowballing is a complementary search strategy [16]. However, there are authors, such as, [17], [18] and [19] who have analyzed the possibility of using snowballing as main search strategy in secondary studies; they advocate the use of snowballing specially in SLRs updates scenario.

TABLE X. SECONDARY STUDIES: SEARCH STRATEGY

Search Strategy	SLR	MS
Automated	70 (59.32%)	24 (46.15%)
Manual	5 (4.24%)	3 (5.77%)
Automated + Manual	32 (27.12%)	13 (25.00%)
Automated + Manual +Snowballing	6 (5.08%)	4 (7.69%)
Automated + Snowballing	5 (4.24%)	8 (15.38%)

RQ4: *Has the quality of the studies been assessed?*

A significant difference between SLRs and SMs refers to study quality evaluation. In SMs the quality assessment is not mandatory, although it may be useful to ensure that there is sufficient information for data extraction. According to [3], performing quality assessment is highly recommended for SLRs, but optional in SMs. Our results, shown in Table XI, confirm this trend, since 62.71% of SLRs had the quality of their studies evaluated and 80.77% of SMs did not evaluate the quality of their included studies.

TABLE XI. SECONDARY STUDIES: QUALITY ASSESSMENT

Study Type	Yes (%)	No (%)
SLR	74 (62.71%)	44 (37.29%)
SM	10 (19.23%)	42 (80.77%)

B. Discussions

In general, the purpose of a secondary study is to provide an overview of a research area and identify research gaps in this area [20]. SLRs and SMs, as secondary studies, enable the identification and aggregation of available evidence to answer research questions [21]. We can affirm that SLRs and SMs support the decision making related to a research to be developed and that there are many similarities between them. However, theoretically, there are also differences, especially regarding the objectives, research questions, search strategy, selection and evaluation of the quality of primary studies, as well as the analysis of data and results obtained. In this study we evaluated if the theoretical differences pointed out in the literature, presented in Section II, reflect in the practical conduction of secondary studies in ES.

Regarding the types of secondary studies that have been conducted by the SE community, SLRs are predominant. One possible explanation is that the first guidelines [1] for conducting SLRs focused on describing this type of study. The SM was superficially presented for the first time, only in 2008, as a mapping process defined by Petersen et al. [22] and later it was updated in 2015 [3]. Therefore, the greater knowledge of the community is on SLRs.

Although secondary studies have been conducted for more than 10 years, only three included studies were updates. Currently, there are processes focused on SLR updates [23], as well as proposals for using visual techniques to support selection of new evidence [24]. However, there is a lack of guidelines on what is updated and how long to update SLRs, explaining the low number of updated studies. It is noteworthy that an outdated secondary study loses its relevance.

Most of secondary studies are published in conferences. One concern with this practice is that there is a limitation of the number of pages available to document the revision. Therefore, it is important that researchers provide additional external information on the conduct of the study in external web pages.

The subareas Software Requirements and Software Engineering Models and Methods concentrated the largest number of secondary studies. As a result it would be interesting the SE community to conduct tertiary studies on these topics.

Petersen et al. [3] affirm that RQs of SLR are specific and RQs of SMs are generic. However, our results showed that both studies presented a predominance of generic RQs. We observed that in this context there is a difference between theory and practice. The community does not distinguish the two studies based on the type of RQ addressed. There is no framework for defining RQs. We can affirm that PICO, although recommended in the guidelines for conducting secondary studies [8], has not been adopted in practice. The set of PICO criteria derived from Medicine [8], therefore, we argue that the main challenge to adopt PICO in an area different from Medicine, is to adapt the set of PICO criteria. For example, Kitchenham and Charters [8] advocate that population could be: (i) a specific role of ES (e.g. tester, manager); (ii) a specific category of software engineer (e.g., novice, experienced); (iii) an application area (e.g., IT systems, control systems); Or (iv) an industry group (e.g. telecom company, small business). Conversely, Biolchini et al. [25] report that population is the group that will be observed by intervention (e.g., publications on the subject investigated).

Surprisingly, in more than 70% of SLRs and SMs researchers do not formally present the search string adopted. As an alert, we can say that the absence of the search string description negatively impacts the reproduction of the secondary studies. In addition to that, both studies (71.27 % SLRs and 70.0 % SMs) have adopted generic strings, containing one or two ANDs (logical operator). Considering the theory, we expected that the SLRs presented strings containing a greater number of ANDs, i.e., more specific strings than SMs. The pilot test to validate the search string is usually not applied, not even in SLRs.

The automatic search, with the support of search strings, has been the main strategy used in both SLRs and SMs.

Corroborating with the theory of Kitchenham and Charters [8] and Petersen et al. [3], most of SLRs (62.71%) evaluate

the quality of their studies, while only 19.23% of SMs adopt the evaluation practice.

As a threat to the validity of this work we highlight:

– Recovery of studies: to avoid missing out relevant studies, one specialist on SLR was asked to create our set of quasi-gold standard. We used this set of studies for filtering the retrieved data set, hence increasing the precision rate of our tertiary study;

– Selection of relevant studies: to validate the selection of studies, an external reviewer independently applied the selection criteria to a subset of randomly studies. Then, we used the inter-raters agreement rate, which was determined using the Cohen's kappa statistic, to measure the reliability of our selection process; – Data extraction: to validate the data extraction activity, an external reviewer independently extracted data from a subset of randomly studies. We intend to further mitigate this threat by making our data set externally available so that our results can be replicated and validated by external researchers.

IV. Conclusions

The main contribution of this paper was to present a practical panorama about the current state-of-the-art on the conduction of secondary studies in SE. To build such panorama, we performed a tertiary study. Based on this panorama, it is possible to affirm that in practice, only the quality assessment is differently conducted in SLRs and SMs.

As future work we intend to: (i) analyze in depth other items to distinguish SLRs from SMs, such as, selection of studies and extraction of data; and (ii) define a decision tree to support SE researchers to decide if conducting an SLR or an SM.

References

[1] B. Kitchenham, "Procedures for performing systematic reviews," Software Engineering Group - Department of Computer Science - Keele University and Empirical SE - National ICT Australia Ltd, Joint Technical Report TR/SE-0401 (Keele) - 0400011T.1 (NICTA), 2004.

[2] B. Kitchenham, D. Budgen, and P. Brereton, *Evidence-Based Software Engineering and Systematic Reviews*, ser. Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series. Chapman & Hall/CRC, 2015.

[3] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: an update," *Information and Software Technology*, vol. 64, no. 1, pp. 1–18, 2015.

[4] H. Zhang and A. Muhammad, "Systematic reviews in software engineering: An empirical investigation," *Information and Software Technology*, vol. 55, no. 7, pp. 1341–1354, 2013.

[5] B. Kitchenham and P. Brereton, "A systematic review of systematic review process research in software engineering," *Information and Software Technology*, vol. 55, no. 12, pp. 2049–2075, 2013.

[6] P. Brereton, B. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *Journal of Systems and Software*, vol. 80, no. 4, pp. 571–583, April 2007.

[7] S. Imtiaz, M. Bano, N. Ikram, and M. Niazi, "A tertiary study: Experiences of conducting systematic literature reviews in software engineering," in *17^{th} International Conference on Evaluation and Assessment in Software Engineering (EASE)*. ACM, 2013, pp. 177–182.

[8] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and Durham University, UK, Tech. Rep. EBSE 2007-001, 2007.

[9] 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.

[10] C. Jia and Y. T. Yu, "Using the 5w+1h model in reporting systematic literature review: A case study on software testing for cloud computing," in *2013 13th International Conference on Quality Software*, July 2013, pp. 222–229.

[11] F. Paz and J. A. Pow-Sang, "Usability evaluation methods for software development: A systematic mapping review," in *2015 8th International Conference on Advanced Software Engineering Its Applications (ASEA)*, Nov 2015, pp. 1–4.

[12] E. Engström, M. Skoglund, and P. Runeson, "Empirical evaluations of regression test selection techniques: A systematic review," in *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '08. New York, NY, USA: ACM, 2008, pp. 22–31. [Online]. Available: http://doi.acm.org/10.1145/1414004.1414011

[13] M. M. Hassan, W. Afzal, M. Blom, B. LindstrÃűm, S. F. Andler, and S. Eldh, "Testability and software robustness: A systematic literature review," in *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, Aug 2015, pp. 341–348.

[14] M. I. Babar, M. Ghazali, and D. N. A. Jawawi, "Systematic reviews in requirements engineering: A systematic review," in *2014 8th. Malaysian Software Engineering Conference (MySEC)*, Sept 2014, pp. 43–48.

[15] M. Riaz, N. Sulayman, M. Salleh, and E. Mendes, "Experiences conducting systematic reviews from novices' perspective," in *14^{th} International Conference on Evaluation and Assessment in Software Engineering (EASE)*. BCS-eWiC, 2010, pp. 1–10.

[16] B. Kitchenham, P. Brereton, M. Turner, M. Niazi, S. Linkman, R. Pretorius, and D. Budgen, "Refining the systematic literature review process–two participant-observer case studies," *Empirical Software Engineering*, vol. 15, no. 6, pp. 618–653, 2010.

[17] S. Jalali and C. Wohlin, "Systematic literature studies: Database searches vs. backward snowballing," in *6^{th} International Symposium on Empirical Software Engineering and Measurement (ESEM)*. ACM, 2012, pp. 29–38.

[18] C. Wohlin, "Second-generation systematic literature studies using snowballing," in *Proceedings of the 20^{th} International Conference on Evaluation and Assessment in Software Engineering (EASE)*. ACM, 2016, pp. 15:1–15:6.

[19] K. Felizardo, E. Mendes, M. Kalinowski, r. F. Souza, and N. Vijaykumar, "Using forward snowballing to update systematic reviews in software engineering," in *10^{th} International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2016.

[20] C. Wohlin, P. Runeson, P. Silveira Neto, E. Engstrom, I. Machado, and E. de Almeida, "On the reliability of mapping studies in software engineering," *Journal of Systems and Software*, vol. 86, no. 10, pp. 2594– 2610, 2013.

[21] B. Kitchenham, R. Pretorius, D. Budgen, P. Brereton, M. Turner, M. Niazi, and S. Linkman, "Systematic literature reviews in software engineering - a tertiary study," *Information and Software Technology*, vol. 52, no. 8, pp. 792–805, 2010.

[22] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *12^{th} International Conference on Evaluation and Assessment in Software Engineering (EASE)*. BCS-eWiC, June 2008, pp. 1–10.

[23] F. Ferrari and J. Maldonado, "Experimenting with a multi-iteration systematic review in software engineering," in *5^{th} Experimental Software Engineering Latin America Workshop (ESELAW'08)*, 2008, pp. 1–10.

[24] K. Felizardo, E. Nakwgawa, S. MacDonell, and J. Maldonado, "A visual analysis approach to update systematic reviews," in *18^{th} International Conference on Evaluation and Assessment in Software Engineering (EASE)*, 2014, pp. 1–10.

[25] J. Biolchini, P. Mian, A. Natali, and G. Travassos, "Systematic review in software engineering: Relevance and utility," PESC/COPPE/UFRJ, Tech. Rep., 2005.

An Experience Report on Update of Systematic Literature Reviews

Lina Garcés
Dept. of Computer Systems
University of São Paulo
São Carlos, SP – Brazil
IRISA - University of South Brittany
Vannes, France
linamgr@icmc.usp.br

Katia R. Felizardo
Department of Computer
Federal Tech. Univ. of Paraná
Cornélio Procópio, PR – Brazil
katiascannavino@utfpr.edu.br

Lucas Bueno R. Oliveira
Federal Inst. of São Paulo
São Carlos, SP – Brazil
lucas.oliveira@ifsp.edu.br

Elisa Yumi Nakagawa
Dept. of Computer Systems
University of São Paulo
São Carlos, SP – Brazil
elisa@icmc.usp.br

***Abstract*—Context: In order to preserve the value of Systematic Literature Reviews (SLRs), they should be frequently updated including new studies produced after the conduction of the reviews. However, most of SLRs are outdated and there is a lack of works that support the conduction of SLRs updates. Objective: The main goal of this paper is to report our experience in updating two of our SLRs. Method: To update these two SLRs, we used automated techniques based on VTM (Visual Text Mining) to guarantee the presence of relevant studies. Results: From our experience, some factors to satisfactorily update SLRs were identified: (i) to adopt software tools to support the updating process; (ii) to provide as much as possible information of previous SLR; (iii) to involve researchers from previous SLR; and (iv) to reuse protocol from preliminar SLR. Conclusions: Reported lessons learned can be used as a basis of knowledge to guide researchers when updating their SLRs.**

***Keywords*—Systematic literature review, update, VTM**

I. Introduction

Evidence Based Software Engineering (EBSE) was first introduced in 2004 as a means of advancing and improving the discipline of Software Engineering (SE) [1]. In this context, Systematic Literature Review (SLR) (a.k.a. Systematic Review (SR)) has provided a methodical, structured process to support the conduction of literature reviews [1] and has then gained substantial importance [2].

If a research area is continually evolving (as it is common in computing), SLRs that are not maintained (i.e., updated) can become out of date. Incorporation of new research or studies (i.e., evidence) into existing SLRs is therefore paramount to sustain their relevance. In other words, SLRs should be frequently updated with the purpose of identifying new evidence that has emerged after the completion of the reviews.

It is worth highlighting that, in the Medicine area, SLRs are in general updated at least every two years, determining whether or not there are new studies available for inclusion in the previous review [3]. In the SE area, the main reasons to update SLRs is that SE professionals and researchers may rely on the results of SLRs to build a body of knowledge about when, how, and what process, techniques, methods, tools, and others are more appropriate to be used. Besides that, SLRs in the SE area have also contributed to identify new, important research topics that have not been treated, yet. Therefore, we argue that the update of SLRs is also a quite important issue in SE.

Even when the same authors update their reviews, search and selection tasks can consume considerable amount of time and efforts, especially if many new studies are obtained during the search, which can lead to difficulties in reading and evaluating evidences. Hence, it may be beneficial to have approaches, including techniques and tools, which support update of SLRs.

Considering that up-to-date SLRs are quite important, but effective update approaches have not been widely investigated in the literature, in our previous paper [4], we presented an automated approach based on Visual Text Mining (VTM) to support the update of SLRs. Such paper also describes an experiment to compare the outcomes achieved using this approach to the ones using traditional (manual) approach. The goal of this paper is to present an experience report on the use of our approach to update SLRs executed by SE researchers. The specific contributions of our work to the body of knowledge in EBSE field are:

1) we bring an experience report by analyzing two real SLRs updated by SE researchers;
2) we present the application of VTM techniques in real SLRs; and
3) we summarized a set of lessons that we have learned when updating SLRs and that could be useful in the context of other SLRs that are intended to be updated.

The remainder of this paper is organized as follows. Section II presents related works and a brief background on how VTM has been used for SLRs. Section III presents the report of our experience in updating SLRs. Section IV presents a brief discussion together with the lessons learned. Finally, conclusions are presented in Section V.

II. Background and Related Works

The process used to extract high-level knowledge from low-level data is known as Knowledge Discovery in Databases (KDD) [5]. Data Mining (DM) is a part of the KDD process

DOI reference number: 10.18293/SEKE2017-078

responsible for extracting patterns or models from data. Visual Data Mining (VDM) is a combination of visualization and traditional DM techniques and is used to explore large datasets [5]. A specific application of VDM, which is of interest in our work, is the amalgamation of text processing algorithms with interactive visualizations to support users to make sense of text collections. By extension, Visual Text Mining (VTM) refers to VDM applied in text or to a collection of documents. According to Paulovich and Minghim [6], the use of VTM can speed up the process of interpreting and extracting useful information from document collections.

Four studies have investigated the use of VTM within the context of EBSE [4], [7], [8], [9].

Malheiros et al. [7] applied content-based VTM techniques to support the selection of primary studies. Similarly, the approach presented by Felizardo et al. [8] also used VTM techniques in the SLR process; however, this approach contains additional visualization techniques based on meta-data analysis. Both works [7], [8] compared the performance of reviewers in carrying out the selection of primary studies by reading abstracts or by using VTM techniques. The works concluded that VTM provides a more precise selection of relevant studies, speeding up the selection task.

Felizardo et al. [9] extended their previous work [8] to support the selection of primary studies, evaluating the decisions of including or excluding primary studies and, mainly, supporting reviewers to ensure as far as possible that important studies are not removed. They concluded that VTM can give solid clues about which particular studies should be checked, reducing the volume of documents that need to be re-evaluated and the time spent in the whole process. VTM therefore seems to be beneficial in supporting the SLR process.

Moreover, VTM has also been beneficial to update SLRs [4] and an approach, called USR–VTM, was proposed and will be also explored in the context of this paper. Because of this, a summary of this approach is presented in next section.

A. USR–VTM Approach

In order to update SLRs using USR–VTM , this approach receives a set of primary studies as input, including [4]: (i) studies included in the previous review; (ii) studies excluded in the previous review; and (iii) studies to be evaluated in the current review (i.e., the new evidence). In order to support selection of new primary studies, two visual representations are built: content-map and edge bundles[1].

A content-map is a two-dimensional visual representation where each m-dimensional instance – a document (primary study) – is mapped on the screen as a graphic element, normally a circle. The process to create a content-map involves the conversion of all primary studies (title, abstract, and keywords) into multi-dimensional vectors. The dimensionality is based on all the terms extracted from the primary studies and it can be reduced eliminating stopwords (i.e., minimally representative terms, such as prepositions, articles, and conjunctions), applying stemming (i.e., converting terms to their radical; for instance, "testing" and "tester" are both reduced to "test") and using projection techniques [10]. In short, a content-map place studies (represented as circles) on the 2D layout in a way that reflects similarity relationships. Thus, similar studies are placed close to one another and dissimilar studies are positioned far apart.

Similarity is calculated using the cosine similarity measure, often used to compare documents in text mining. It is a measure of similarity between two vectors of *n* dimensions by finding the cosine of the angle between them, which ranges between 0 (no similarity) and 1 (completely similar) [11]. The content-map presented in Figure 1 contains studies included (green points) and excluded (red points), in a previous SLR, as well as studies to be evaluated (grey points) in the update of such review. Primary studies are connected with their neighbours by applying KNN (K-Nearest Neighbour) Edges Connection technique [12]. This technique connects nodes with their nearest neighbours, computing the proximity on the projection itself.

Another visual representation is edge bundles, which is a hierarchical tree visualization technique that shows both nodes and node-links (relationships between nodes) [13]. In the case of an SLR, nodes (small circles, as shown in Figure 1) are the primary studies (i.e., studies included in the previous review, studies excluded in the previous review, and studies to be evaluated in the current review) and node-links (blue lines, also shown in Figure 1), are the citations between them.

In order to create edge bundles, the HiPP (Hierarchical Point Placement) strategy [6] is used. In the edge bundles view, node-links were coloured to represent the direction of the citation: the citing study is at the light blue end of the link and the cited study at the dark blue end. In summary, analyzing the edge bundles it is possible to identify the number of times that a study has been cited by other studies. Notice that studies represented in the content-map are the same ones in the edge bundles; i.e., each study in the content-map has a corresponding study in the edge bundles.

Revis[2] tool was used to create these visual representations. It takes only seconds to create and present these views with a few hundred documents.

Two strategies to include and exclude new primary studies using both content-map and edge bundles can be applied to support selection of studies to update SLRs [4]:

- **Inclusion Strategy:** To include a new primary study, it must be a neighbour of at least one previously included study (observed in the content-map) **AND** it must not cite previously excluded paper(s) (observed in the edge bundles). Neighbours are nodes connected through edges between them; and
- **Exclusion Strategy:** To exclude a primary study, it must be neighbour of only previously excluded studies (or studies in evaluation) **AND** it must not cite previously included papers.

[1] In general, visualization techniques use colours to add extra information on a visual representation; hence, we recommend reading of a colour version of this paper.

[2] Revis is a visualization and interaction tool that offers Visual Text Mining exploration of document collections. It is freely available at https://www.dropbox.com/sh/2ahjt6urucm114s/AAD8GF3WmNAtYrvHbzOKzgeqa?dl=0

Undefined or unclear situations should be given particular attention. For example, a study may be a neighbour of previously excluded studies; however, it cites previously included study(ies), i.e., the New Evidence – NE (grey points) are linked to previously excluded study(ies) (red point(s) in the content-map) and cite previously included study(ies) (green point(s) in the edge bundles). In a situation such as this, the new primary study should be analyzed by reviewers.

By exploiting the strong visual processing abilities of humans, the USR–VTM approach is an important ally to be used during update of SLRs, facilitating and enhancing interpretation and decision-making in regard to the selection of new primary studies. USR–VTM can give solid clues (except to undefined situations) about which particular studies should be included or excluded. However, the final decision is taken by researchers (experts in the SLR domain). In the following, we present our experiences in updating two SLRs using the two strategies described bellow for selection of new studies.

III. Experience Report

This section presents our experiences in updating two SLRs. The first one (hereafter referred to as SLR1) had the goal of finding the state of the art about Reference Architectures and Reference Models for Ambient Assisted Living (AAL) [14]. The second one (referred to as SLR2) was focused on works about development of service-oriented robotic systems [15]. It is observed that the first version of both SLR1 and SLR2 had been conducted at least two year ago, then an update becomes necessary. In next sections, we discuss our experiences in updating both SLRs (referred as SLR1' and SLR2').

A. Updating SLR on Reference Architectures and Reference Models in AAL

To update SLR1, we revisited its protocol that contains all required information to execute an SLR (e.g., research questions, search string, search databases, among others). It is worth saying that only the period to be updated was changed. Following, we conducted the searches in the same databases used in SLR1 (namely, ACM Digital Library, IEEE xplore, Springer, ScienceDirect, Compendex, Scopus, and Web of Science) and also using the same search string. All primary studies were managed using JabRef[3], which is an open source bibliography reference manager tool. We exported information of all primary studies found (i.e., title, author(s), abstract, keywords, year of publication, and the name of the data source) to JabRef. For databases that do not support JabRef, we manually get such information. During the searches, we considered studies published after November, 2015. We needed to remove repeated studies, as different databases can sometimes find the same studies. As a result, a total of 84 non-repeated studies were obtained.

Aiming to select the primary studies to be in fact included in SLR1', two researchers performed independently a brief reading only considering titles and abstracts of the studies, intending to exclude studies far away from the topic of this SLR. A total of 23 studies (27.4% of 84 studies) were considered to be further analyzed. After that, we used the techniques based on VTM described in Section II-A. Hence, we applied USR–VTM that generated content-map and edge bundles for this update, as showed in Figure 1. As mentioned previously, the 14 studies included in SLR1 are coloured in green, the 50 studies excluded in SLR1 are coloured in red, and the 23 studies to be analyzed are coloured in grey. USR–VTM was then applied to analyze the 23 new studies. For sake of space limitation, Figure 1 exemplifies three of them (NE[17], NE[18], and NE[20]).

As suggested by the Inclusion Strategy of USR–VTM, NE[17] is neighbour of at least one previously included study (green points in the content-map) **and** it do not cite previously excluded papers (red points in the edge bundles). When an NE is allocated in the content-map next to other included papers, there is a strong indication that this NE should be included. It is also possible to observe that NE[17] indeed do not share citations with excluded papers (red points). In both views (content-map and edge bundles), we had clues that NE[17] is relevant for our review and that it should be included. Therefore, our final decision was to include it.

The second study is NE[18] (pointed out in Figure 1), which is neighbour of previously excluded studies; however, it cites previously included studies. NE[18] was classified as an undefined situation by USR–VTM. In this case, this study was read by us and the final decision was to excluded it. This study deserved special attention.

The third study is NE[20] (pointed out in Figure 1). As suggested by the Exclusion Strategy of USR–VTM, NE[20] is neighbour of previously excluded studies (red points in the content-map) **and** it is not linked to previously included papers (green points in the edge bundles). On the other hand, NE[20] is linked to one previously excluded paper. Therefore, the views encouraged us to exclude NE[20].

After applying the Inclusion/Exclusion Strategies in all 23 studies, including those ones exemplified above, we selected seven new studies to be included, resulting in a total of 21 relevant studies for this SLR (14 from SLR1 plus seven new studies). After that, the full text of these seven studies was read and the required data for updating the answers of the research questions were extracted. For this, the data extraction form created to SLR1 was used without changes. For each new included study, this form was filled by one researcher. For validation purposes, a sample comprising 30% of the total number of primary studies was selected randomly and had their data extracted by other researchers. Whenever the data extracted differed, differences was discussed until consensus was reached. The data synthesis involved the compilation of the data extracted from each new primary study and merge of such data into the previous answers for each research question. Besides that, in the previous version of this SLR, the data synthesis for the research questions was performed using a quantitative table. Reuse of same table facilitated the analysis during update. Finally, it is also worth highlighting that SLR1 and SLR1' were conducted by the same researchers.

B. Updating SLR on Robotics

To update SLR2, initially, its protocol was revisited to identify required changes. The research questions were not changed, but the search strategy was modified to restrict the

[3]http://www.jabref.org/

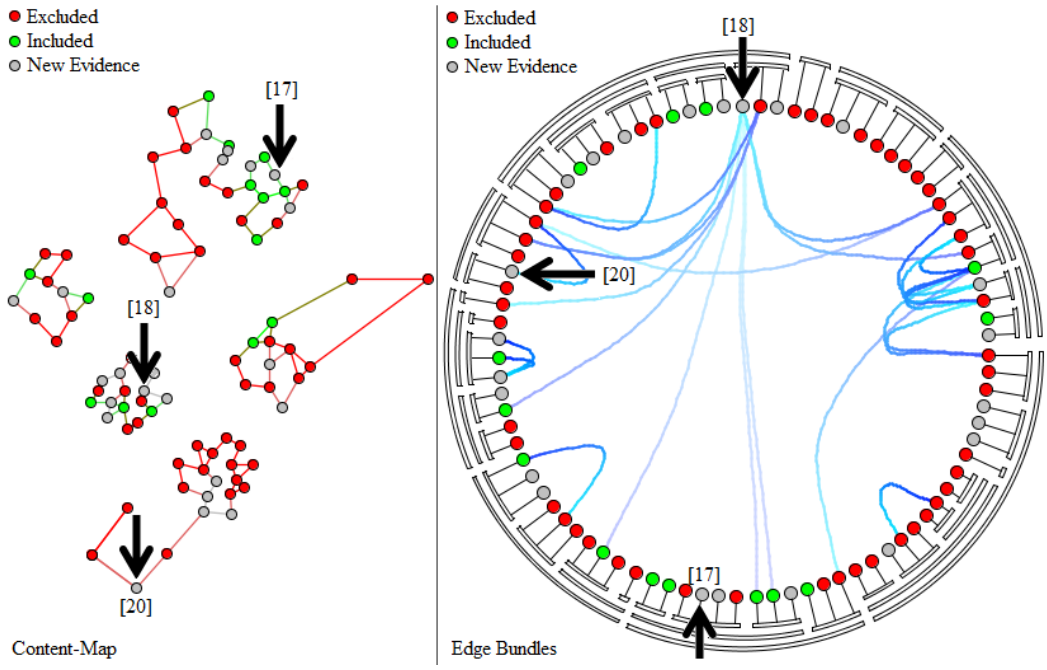


Fig. 1. Views to support the update of SLR1.

search only for studies published after 2011, avoiding overlap among studies retrieved in SLR2 and SLR2'.

To find new primary studies, we used the same databases used in SLR2. New terms to the search string were not identified, but minor changes were required because during 2014, some search engines were modified, demanding adaptations of search strings. After changing the protocol, it was tested to verify its feasibility and adequacy based on a pre-selected set of relevant studies. A total of 158 studies were retrieved from the search engines. Following, repetitive studies were removed and after a brief reading in the title and abstract of each study, we selected 28 studies that could be relevant to this SLR.

Similarly to SLR1', the study selection activity of SLR2' was performed with the support of USR–VTM and its views. Figure 2 shows the content-map and edge bundles related to this SLR. As mentioned previously, in these views, the 39 studies included in SLR2 are coloured in green, the 36 studies excluded in SLR2 are coloured in red, and the 28 studies to be analyzed are coloured in grey. These views were generated in the same way described in Section III-A. In the sequence, we illustrate the application of USR–VTM in three of the 28 studies (NE[5], NE[14], and NE[26]).

NE[5] (pointed out in Figure 2) is the first example. As suggested by Inclusion Strategy of USR–VTM, NE[5] is neighbour of studies in evaluation) **and** it must not cite previously included papers. Consequently, we decided to exclude it.

NE[14] (pointed out in Figure 2) is neighbour of at least one previously included study **and** it does not cite previously excluded papers. On the other hand, NE[14] cites four previously included papers. This fact might be taken as a strong indication of NE[14] is a relevant study, indicating its inclusion. Therefore, our final decision was to include it.

NE[26] (pointed out in Figure 2) is a typical example of a study difficult to be selected, because it is neighbour of previously excluded studies and it cites three previously included studies. This study was carefully analyzed by us. We decided by its inclusion. As a result, 18 primary studies were then selected for full reading and data extraction.

With the final set of primary studies decided upon, the data extraction activity was carried out on all 18 studies that passed the screening process. The information for answering each research question was tabulated using themes, which were defined by one researcher. Thereafter, studies belonging to each theme were counted. This researcher was also responsible to extract the data and complete the data extraction form. Results were summarized to present an overview of the findings. The data synthesis were performed using tables (showing the totals and summaries).

IV. Discussions

The two case studies presented herein were carried out aiming to investigate and evaluate the use of USR–VTM for updating real SLRs involving SE researchers. The main advantage of USR–VTM is that the selection activity (classification of studies as included or excluded) is based on previous knowledge. For example, if a study was considered relevant (i.e., it was included) in a previous SLR, and a new study is similar in terms of content to this study, then it is a clue that the new study could be included. Similarly, if a new study cites or is cited by previous relevant studies, then it could be included. On the other hand, a new study that is dissimilar in terms of content of all previous included studies can probably be excluded. In addition, a new study that cites or is cited by previous excluded studies has a chance to be irrelevant to the scope of the systematic review. As USR–VTM is new in EBSE field, for a while, it is not our intention to recommend the elimination of the traditional selection approach

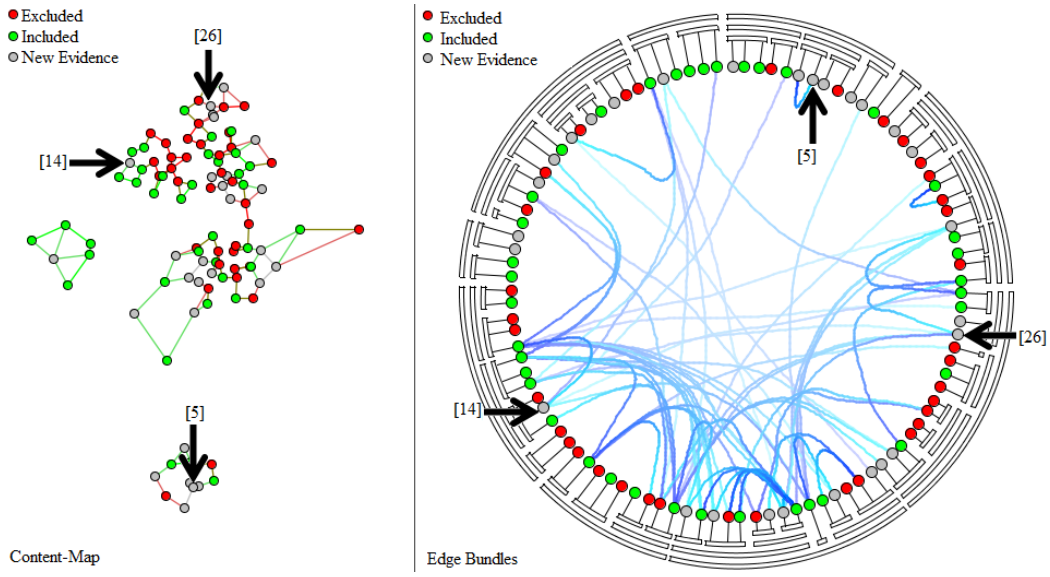


Fig. 2. Views to support the update of SLR2.

(title/abstracts/full text reading). USR–VTM intends to be a complementary approach that can be used in conjunction with the traditional one, mainly to speed up the selection activity. In this perspective, USR–VTM could be also used before or after applying the traditional approach for selecting new primary studies, aiming to be a double-checking mechanism.

Considering our experience with regard to both updates and other SLRs that we have updated in the last years, following, we discuss four main lessons learned:

- Lesson 1 – Choose tools to support updates: When researchers are conducting an SLR, selection of relevant studies can consume considerable time and effort. In particular, the obtaining of many studies during the search task leads to difficulties in reading and evaluating the set of studies. To update SLRs, adopting a supporting software tools is also quite important to mainly facilitate primary studies management during the selection of new studies. They become even more important when they store primary studies found in the previous version of the SLR and information of these studies will be used to update. When it is intended to apply an approach like USR–VTM[4], supporting tooling together with data of primary studies (e.g., LaTeX *.bib* files) are essential. We adopted Jabref tool to support both conduction and update of SLRs.
- Lesson 2 – Keep information of previous SLRs: If it is intended to conduct an update of an SLR, it is essential to keep all information related to that SLR, especially the set of studies returned from each database, the adapted search string for each database, and even the set of excluded studies. In particular, as presented in our experiences in this paper, when approaches like USR–VTM are intended to be used during update, previously studies included and excluded are required (i.e., it is necessary to register and make available the *.bib* files containing both set of studies). Furthermore, previous information is a rich source for future checking. For example, it can answer doubts, such as which were the reasons to a given study to have been included/excluded and who included/excluded it, why those databases were selected and used, why some terms were not considered in the search string, and so on. While this lesson is of course valuable, we claim that it is usually underrated by EBSE researchers. We recommend to EBSE researchers that they make available such type of information on their SLRs, instead of only the final report.
- Lesson 3 – Form a team containing a researcher who has already participated in the previous review: We argue that the effort to update an SLR is significantly higher if none of the researchers of the previous review as part of the team, providing direct access to the employed instruments. In spite of the SLR documentation should make it possible to be updated by any team of researchers, there is a tacit knowledge (in the mind of researchers) that the documentation related to SLRs do not mention. There is an amount of implicit knowledge (that we refer as "global scenario" of the research topic of the SLR) acquired from a large number of primary studies usually found in the previous execution of SLRs. It is interesting to say that this global scenario contains knowledge from both included and excluded studies, as well as the entire process and tasks by which the SLR was carried out. Therefore, the direct gain using the same (or part of the same) team is take advantages of the global scenario in the mind of researchers. There is therefore a faster access to information (even documented information), while new teams also need to understand how the information is organized.
  For instance, considering our experience regarding SLR1/SLR1', the global scenario was formed from 273 studies analyzed in the first conduction. As we used the same team during update, this global scenario was fundamental to analyze other 84 studies during

update. Otherwise, if new teams updated it, these 84 studies needed to be analyzed without any support of previous acquired knowledge.
It is worth also highlighting that there is a lack of concrete studies proposing or discussing about the ideal researchers team (in terms of number of researchers and their skills) for both SLR conduction and update. However, from our experiences in conducting and updating SLRs, teams should include at least three researchers: two experts in the research topic being explored in the SLR (at least one being a high experienced researcher) and one specialist in SLR. It is also important that all team members are familiar with conduction of SLRs and have experience in using supporting tools required for SLRs, e.g., Revis and JabRef.

- Lesson 4 – Reuse SLR protocol: As well known, the establishment of the SLR protocol usually consumes a considerable amount of time and effort, besides being a complex task. However, it is essential to assure the quality of SLRs. Reusing such protocol is undoubtedly interesting when updating SLRs. In our two experiences, We replicated the original protocols. In this context, we experienced the importance of having access to a complete and detailed protocol to guide our updates. The time spent in checking SLR protocols was quite reduced if compared to protocol preparation to the first conductions. Only considering our two SLRs, the time consumed to check the protocol was less than 1/5 of the time consumed to prepare a protocol from scratch. Therefore, reusing the protocol is in fact quite relevant.

V. Conclusions and Future Work

Since knowledge in any research area continually evolves, SLRs that intend to represent such knowledge must be also from time to time updated. Otherwise, their validity can be undetermined. In this scenario, when SLRs are updated, considerable efforts must be applied; besides, no guidelines on how to conduct such updates are available. The main contribution of this work was to report experiences in updating two SLRs and also report lessons learned, thus providing a basis for other researchers who intend to update SRLs.

We also showed that the use of USR-VTM approach can support and speed up the SLR updates. This work has helped us to understand the usefulness of VTM techniques to support those updates. We observed that USR–VTM can support tasks that involve large collections of data, such as the studies collected and to be evaluated. Through these two experiences and others, we argue that adoption of supporting tools, like that one automating USR–VTM, brings important advantages, mainly facilitating the selection of new studies and also becoming results of SLR updates more trustworthy. In a scenario of a team of reviewers conducting an SLR, as the case of SLR1' and SLR2', USR–VTM can be considered a valuable means to reach the conclusion on what should and should not be included. The views supported the team to reach a common sense about inclusions and exclusions. On the other hand, in the special case of an SLR executed by only one reviewer, this researcher should consider discussing his or her decisions with other researchers. Alternatively, the researcher could use our approach as the external opinion. Moreover, by exploring different visual representations of primary studies through USR–VTM' views, researchers have additional and complementary detailed information, for example, citations between primary studies, which is not readily available only reading studies.

For future work, we intend to use USR–VTM to also update Systematic Mappings (SM), considering that selection of new primary studies is quite similar in both SM and SLR; therefore, we believe our approach is also suitable to it.

Acknowledgments This work is supported by FAPESP (Grants N.: 2013/20317-9, 2014/02244-7, and 2015/19192-2).

References

[1] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and Durham University, UK, Tech. Rep. EBSE 2007-001, 2007.

[2] H. Zhang, B. Muhammad, and T. Paolo, "Identifying relevant studies in software engineering," *Information and Software Technology*, vol. 53, no. 6, pp. 625–637, 2011.

[3] J. Higgins, S. Green, and R. Scholten, *Maintaining Reviews: Updates, Amendments and Feedback*, 2008, pp. 31–49.

[4] E. Felizardo, K.R. ad Nakagawa, S. MacDonell, and J. Maldonado, "A visual analysis approach to update systematic reviews," in *18^{th} Int. Conference on Evaluation and Assessment in Software Engineering (EASE'14)*, 2014, pp. 4:1–4:10.

[5] D. Keim, "Information visualization and visual data mining," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 1–8, 2002.

[6] F. Paulovich and R. Minghim, "Hipp: A novel hierarchical point placement strategy and its application to the exploration of document collections," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1229–1236, 2008.

[7] V. Malheiros, E. Höhn, R. Pinho, M. Mendonça, and J. Maldonado, "A visual text mining approach for systematic reviews," in *1^{st} Int. Symposium on Empirical Software Engineering and Measurement (ESEM'07)*, 2007, pp. 245–254.

[8] K. Felizardo, N. Salleh, R. Martins, E. Mendes, S. MacDonell, and J. Maldonado, "Using visual text mining to support the study selection activity in systematic literature reviews," in *5^{th} Int. Symposium on Empirical Software Engineering and Measurement (ESEM'11)*, 2011, pp. 1–10.

[9] K. Felizardo, G. Andery, F. Paulovich, R. Minghim, and J. Maldonado, "A visual analysis approach to validate the selection review of primary studies in systematic reviews," *Information and Software Technology*, vol. 54, no. 10, pp. 1079–1091, 2012.

[10] F. V. Paulovich, L. G. Nonato, R. Minghim, and H. Levkowitz, "Least squares projection: a fast high precision multidimensional projection technique and its application to document mapping," *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 3, pp. 364–375, 2008.

[11] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, 1st ed. Pearson, 2014.

[12] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[13] D. Holten, "Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 741–748, 2006.

[14] L. Garcés, A. Ampatzoglou, F. Oquendo, and E. Y. Nakagawa, "Reference architectures and reference models for ambient assisted living systems: A systematic literature review," University of São Paulo (USP), Tech. Rep. No. 414. 1–32, 2017.

[15] L. B. R. Oliveira, "Architectural design of service-oriented robotic systems," Ph.D. dissertation, University of São Paulo (USP) / Université de Bretagne-Sud (UBS), São Carlos, Brazil / Vannes, France, 2015.

System of Systems Requirements: A Systematic Literature Review using Snowballing

Renata Martinuzzi de Lima [1], Daniel de Vargas [1], Lisandra Manzoni Fontoura [1]
[1] Universidade Federal de Santa Maria (UFSM) - Santa Maria, RS, Brazil
{rlima, ddvargas, lisandra}@inf.ufsm.br

***Abstract* - System of Systems (SoS) is a subject of interest for many researchers from different research lines. System of Systems Engineering (SoSE) deals with many challenges mainly in the activities related to requirements. In order to explore studies related to Requirements Engineering (RE) for SoS, this Systematic Literature Review, using the Snowballing approach, aims to answer four research questions. These questions are related to the activities of translating capability objectives in SoS requirements, SoS requirements specification, SoS requirements management during the evolutionary process of a SoS, and as the main goal, understanding the research challenges of RE for SoS which still need to be explored.**

Keywords - System of Systems; Requirements Engineering; Capability Objectives; Research challenges.

I. INTRODUCTION

In different application domains, it is possible to find distinct systems working together to satisfy a specific goal. It represents what is named as "System of Systems" (SoS). System of Systems Engineering (SoSE) is a subject of increasing interest within the Systems Engineering (SE) community. However, structural differences, the diversity of stakeholders in a SoS development project leads to a complexities and challenges still unexplored in SE [7].

Requirements Engineering (RE) is one of the most critical and difficult tasks in the development of any kind of system. Therefore, in SoS context the complexity, problems and challenges increase. The traditional RE for individual system is still a big challenge for system engineers and managers, but it is already quite explored and there are many techniques proposed with success. However, in the context of SoS Engineering, these techniques from SE not always can be perfectly applied to SoSE. Hence, it is necessary studies to develop new methods, techniques or processes specifically directed to address demands on requirements for SoS [5].

To begin new studies in this topic, it is interesting an investigation about how the subject has been treated by researchers in the last years. Many reviews and surveys can be found in the literature of SoS field, however there is no evidence on Systematic Literature Reviews (SLR) or Systematic Mapping Reviews (SMR) specific on Requirements for SoS. Therefore, the main goal of this SLR is to explore studies related to RE for SoS, using the Snowballing approach, in order to find papers that answer four research questions.

This paper presents the results of a SLR in studies published from 2005 to 2016 and it was conducted following the Snowballing approach. The section II presents a background about Requirements Engineering for SoS; Section III has the details about the method used in the SLR; Section IV presents the results and Section V the conclusions.

II. BACKGROUND

A classic definition, given by [6], says that System of Systems (SoS) is a collection of components that individually must be recognized as systems and may have two fundamental characteristics: i) Operational Independence of Components and ii) Managerial Independence of Components. In addition, it defines six main features of a SoS: Operational Independence, Managerial Independence, Geographic Distribution, Evolutionary Development and Emergent Behavior [6].

There are seven core elements of SoS Engineering (SoSE), defined by [3], which deal with the application of the SE processes in the SoS context. Three of these seven core elements are directly related to requirements: "Translating Capability Objectives", "Developing and Evolving a SoS Architecture", and "Addressing Requirements and Solution Options". The first one describes one of the most important tasks of SoSE that aims to develop an initial understanding of the capability objectives, which may be stated in high level and based on desired operational tasks or missions.

Traditional Systems Engineering (SE) has its methods and processes well defined, thus the elicitation and definition of requirements becomes a trivial task. However, assuming that SoSE is not only a extension of SE, it is true to say that SoS requirements definition practices are also not a simple extension of traditional RE [5]. In this sense, RE for SoS involves the development of the tasks of "translating capability objectives into requirements", as well as the use of methods, tools, and processes that support this task. Also, the management of requirements throughout the development and evolution of a SoS.

III. METHODOLOGY

SLR is a specific research methodology that is developed with the purpose of gathering and evaluating studies in a particular topic of interest and using a specific approach [1]. The snowballing approach is a way to develop a SLR using the reference list and the citations of a specific paper in order

DOI reference number: 10.18293/SEKE2017-114

to identify additional studies about that subject, looking at where papers are actually referenced and cited [8].

The planning phase of a SLR includes the formulation of the research questions, the definition of the approach that will be used, and the definition of the inclusion and exclusion criteria [4]. In this SLR, the research questions are the following:

(RQ1) Which papers are related to the translation of capability objectives in high-level SoS requirements?

(RQ2) Is there some way commonly used to specify SoS requirements?

(RQ3) Which papers are related to the management of the requirements changes in the evolutionary process of a SoS?

(RQ4) What are the research challenges found in Requirements Engineering for SoS?

In addition, in the snowballing approach, it is necessary to define a tentative of Start Set, that is an initial set of papers which will be analyzed in the backward and forward snowballing. The first attempt of a start set has been defined based on studies that were used in a previously research.

After that, it has been defined the inclusion and exclusion criteria, in the context of the SLR. For inclusion: the paper should have been published between 2005 and 2016; the title, keywords, or abstract should make explicit that the paper is related to the research topic; and the paper should answer, in some way, at least one of the research questions. For exclusion: The publication should not be a tutorial, workshop, only an abstract or a technical report; the full text should be available; and redundant publications should be excluded.

The second phase begins with the start set. In the first interaction, the backward and forward snowballing analyze each paper. The papers included in the first interaction are analyzed in the second interaction and so on. Backward Snowballing analyzes the reference list to identify new papers to include in the SLR. First, the papers are excluded based on the exclusion criteria cited above, and then, based on title, abstract, and conclusion, candidates papers are reserved to be more carefully analyzed [8]. Forward Snowballing refers to identifying new papers to include by analyzing the list of papers citing the paper being examined. Using the Google Scholar search tool, the first screening is done based only on the information available on Google. If this information is not sufficient to decide whether the paper is a candidate or not, it is analyzed more carefully applying the exclusion and inclusion criteria [8].

The initial start set had 5 papers identified by [A1] to [A5] to begin the first interaction. In the first interaction, after backward and forward snowballing, 28 candidates were identified and 7 papers were included ([A6] to [A12]). In the second interaction, 7 candidates were examined and 2 papers were included ([A13] and [A14]). In the third interaction, from 4 candidates, 3 papers were included ([A15] to [A17]). In the fourth interaction, after examining 3 candidates, more 2 papers were included ([A18] and [A19]). Since after the fifth interaction no new paper has been identified, the loop is finished and the data extraction phase could be initiated. All the papers included are identified on Table II.

Data extraction was done through the Reading Sheets that gather information such as: title, authors, year, country and place of publication, the main research topic, objective, type of article, application context, as well as the relevant information that answers the research questions.

IV. RESULTS

At the end of the execution phase, 19 papers met the inclusion criteria and their data were extracted. Table II shows the details of all the papers included in the SLR with their IDs, titles, authors and year of publication. We also present the calculation of the efficiency and the citation matrix of the Snowballing approach, both suggested by [8], and then the results according to each Research Question.

A. Efficiency

An important efficiency measure for Systematic Literature Reviews is to analyze the number of papers included in relation to the total number of candidates examined. WOHLIN [8] emphasizes that efficiency is calculated based on the candidates. For example, if we remove those candidates from backward snowballing where the exclusion decision is taken by the year of publication or title in the reference list, efficiency increases. In forward snowballing, the analysis is done in the same way, based on the observation of the data found in Google Scholar, so efficiency can increase in general.

Therefore, the efficiency of this work (24%) can be considered high because the number of candidates is low due to the research topic is very limited. Thus, the analysis of the efficiency in the different interactions is shown in Table I.

TABLE I. EFFICIENCY

Interaction	***Candidates***	***Included***	***Efficiency***
Start Set	7	5	5/7 = 71%
First Interaction	23	7	7/23 = 30%
Second Interaction	6	2	2/6 = 33%
Third Interaction	4	3	3/4 = 75%
Fourth Interaction	3	2	2/3 = 66%
Fifth Interaction	0	0	0
Total = 24% of efficiency			

B. Citation Matrix

To understand the references and citations among the included papers, a citation matrix was created, based on [8] suggestion. Table III shows how the 19 papers reference to each other, denoted by an "x" (e.g. paper [A2] references to papers [A3] and [A4] and it is cited by [A11]). Table III is complemented with information on the possibility of citation, denoted by a "-". For example, paper [A4] does not mention any other of the included papers, since by checking the year of publication it can be seen that papers marked with "-" could not be referenced because they have not yet been published. However, cells left in blank refer to papers that were published prior to [A4] and could have referenced them.

C. RQ 1- Which papers are related to the translation of capability objectives in high-level SoS requirements?

[A3] defines three core elements of SoSE related to requirements: "Translating Capability Objectives," "Developing and Evolving an Architecture", and "Meeting Solution Requirements and Options." [A8] provides guidance for defining capability objectives, it defines SoS

Engineering methods, processes and tools tailored to support this activity. Finally, [A15] proposes a capability-based approach that mathematically explores the structural semantics of representing user needs for formulating requirements in complex systems.

TABLE II. ALL THE PAPERS INCLUDED IN THE SLR

ID	Authors	Year	Title
[A1]	BOARDMAN, John; SAUSER, Brian	2006	System of Systems - the meaning of of.
[A2]	DAHMANN, J. S. et al	2010	Systems Engineering Artifacts for SoS.
[A3]	DoD-USA	2008	Systems Engineering Guide for Systems of Systems.
[A4]	DAHMANN, Judith S.; BALDWIN, Kristen J	2008	Understanding the current state of US defense systems of systems and their implications for systems engineering.
[A5]	NIELSEN, Claus Ballegaard et al	2015	SoSE: basic concepts, model-based techniques, and research directions.
[A6]	LOCK, Russell	2012	Developing a methodology to support the evolution of SoS using risk analysis.
[A7]	CECCARELLI, Andrea et al	2015	Introducing Meta-Requirements for Describing System of Systems.
[A8]	LANE, Jo Ann	2014	System of systems capability to requirements engineering.
[A9]	HOLT, Jon et al	2015	A model-based approach for requirements engineering for systems of systems.
[A10]	VIERHAUSER, Michael et al	2015	A requirements monitoring model for systems of systems.
[A11]	HALLERSTEDE, Stefan et al	2012	Technical challenges of SoS requirements engineering.
[A12]	HOLT, Jon et al	2012	Model-based requirements engineering for system of systems.
[A13]	LEWIS, Grace A. et al	2009	Requirements engineering for systems of systems.
[A14]	KEATING, Charles B.; PADILLA, J.; ADAMS, K.	2008	System of systems engineering requirements: challenges and guidelines.
[A15]	RAVICHANDAR, R.; ARTHUR, J. D.; BOHNER, S.	2007	Capabilities engineering: Constructing change-tolerant systems.
[A16]	MACDIARMID, Alisdair; LINDSAY, Peter	2010	Can system of systems be given self-x requirement engineering capabilities?
[A17]	WALKER, Randy G.; KEATING, Charles B	2012	Defining SoS requirements: an early glimpse at a methodology.
[A18]	KATINA, Polinpapilinho F.; KEATING, Charles B.; RA'ED, M. Jaradat	2014	System requirements engineering in complex situations.
[A19]	WALKER, Randy G. et al	2014	A method to define SoS requirements.

TABLE III. CITATION MATRIX

	Citation																		
Ref.	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**	**A11**	**A12**	**A13**	**A14**	**A15**	**A16**	**A17**	**A18**	**A19**
A1		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
A2			X	X	-	-	-	-	-	-	-	-				-	-	-	
A3		-		X	-	-	-	-	-	-	-	-	-	-		-	-	-	
A4		-	-		-	-	-	-	-	-	-	-	-	-		-	-	-	
A5	X		X	X			-		-	-	X	X							
A6	X				-		-	-	-	-	-	-					-	-	
A7	X			X	-				-	-									
A8			X		-		-		-	-								-	
A9			X	X	-		-			-		X	X						
A10				X	-		-		-					X					
A11		X			-	-	-	-	-	-		-	X				-	-	
A12			X		-	-	-	-	-	-	-		X				-	-	
A13		-	X		-	-	-	-	-	-	-	-				-	-	-	
A14		-	-	-	-	-	-	-	-	-	-	-	-			-	-	-	
A15		-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	
A16		-			-	-	-	-	-	-	-	-	X	X			-	-	
A17			X		-	-	-	-	-	-	-	-		X				X	
A18			X		-		-	-	-	-				X					
A19			X		-		-	-	-	-				X				X	

D. *RQ 2 - Is there some way commonly used to specify SoS requirements?*

[A2] identifies 14 ESoS artifacts, including requirements-related artifacts: "Capability Objectives", "CONOPS", "Information about Systems That Impact SOS Capability Objectives" and "Requirements Space". Meanwhile, [A7] proposes a model of meta-requirements to describe points of view of a SoS and relate them. This model was developed to be used in the derivation of requirements of any SoS.

[A9] is an evolution of the research presented in [A12] and both of them present a model-based approach to RE for SoS which represents the best practices is SoS in terms of proven standards and research derived from a model-based SE approach to requirements engineering. Another RE method for SoS is presented by [A13], and this brings the following top-down and bottom-up activities: identify SoS context, identify SoS and individual system goals, understand SoS interactions, identify system capabilities and constraints and analyze the gap.

[A14] provides a guideline within SoSE efforts exploring the nature of requirements from a SoSE perspective and establishing a foundation for differences between the SE and SoSE problem domains. Moreover, [A16] explores a vision of how the key artifacts of requirements engineering need to evolve with supporting tools and processes in order to support the development of a SoS, it also lists the four types of SoS and illustrates their relationship among RE and each of them.

[A18] presents an alternative perspective for systems thinking-based and an approach for requirements elicitation in complex situations, the case of a SoS, exploring some broad challenges associated with the requirements engineering elicitation.

The same authors published [A17] and [A19]. While [A17] brings a methodology proposal, still in development, that combines top-down and bottom-up approaches to the derivation of SoS requirements, [A19] provides an update to the [A17], now renaming the "methodology" by "method".

E. RQ 3 - Which papers are related to the management of the requirements changes in the evolutionary process of a SoS?

[A1] points out important characteristics that must be considered on the requirements management in the evolutionary process. While [A3] introduces the core elements: "Developing and Evolving a SoS Architecture" and "Requirements management" which are related to the changing management in the evolutionary process of a SoS.

[A6] proposes a methodology to support the identification, organization and discussion of needed information to manage the evolution of a SoS in terms of requirements. Meanwhile, [A10] describes a three-dimensional requirements monitoring model for SoS, an essential task after updating certain components or constituent systems in the evolutionary process.

Then, [A15] proposes an alternative approach to developing complex and emerging systems that need to be change tolerant, considering that they have long development cycles. Also, [A16] suggests that managing the requirements evolution must be an autonomous process.

F. RQ 4 - What are the research challenges found in Requirements Engineering for SoS?

Paper [A3] concludes that some SoSE issues, particularly in the military field, still need to be addressed including options for managing SoS that would facilitate SoSE and ensure more predictable progress, effective ways of achieving SoS evolution, strategies to effectively integrate constituent systems into a viable, evolving, and in some cases ah-hoc SoS.

[A4] suggests some areas for further investigation into SoS management, also in the military field, especially on the existing need to clarify the management relationships between SoS and constituent systems. Moreover, in the technical context, the authors cite, as an important role of SoSE, the need to create mechanisms that can anticipate the changes in the requirements and evaluate the implications of these changes with the constituent systems managers and engineers.

Paper [A5] identifies that the need for interoperability in a SoS adds some important requirements in terms of analysis and modeling methods, which increases a need for techniques that support a verification of requirements that serve the constituent systems. Also, [A5] states that many system-level tests fail because the emerging proprieties were insufficiently captured during the requirements development phase.

[A7] identifies some challenges from the perspective of SoSE requirements analysis. It highlights the following assumptions: i) "The requirements of time, reliability, and security can become more complex when the focus is on SoS as a whole". ii) "As a SoS is an evolutionary system, adaptability and flexibility should be considered in the short and long term, in which case specific SoSE approaches should be reviewed". iii) "A set of SoS requirements requires an understanding of a potentially large number of scenarios, where boundaries can be difficult to define, to be able to build a SoS, engineers must understand where to set boundaries and how large and detailed their view of SoS must be".

The paper [A11] lists the main technical activities of requirements, validation, tracing and verification processes with the main characteristics of a SoS (independence, distribution, emergency and evolution) which allows to identify several technical challenges in RE for SoS.

[A13] discusses some RE for SoS challenges in the following contexts: scale, multiple domains, varied operational context, decentralized control, fast evolving environments, continuous and disconnected execution of the different phases of the life cycle and the needs for collaboration and integration.

[A14] indicates key implications for the development and use of requirements in SoSE in practice. (e.g. the nature of the SoSE domain problem suggests that requirements are simultaneously "loose and tight"; the resolution of requirements should increase with further understanding of the SoS domain problem and emerging conditions; SoS requirements and constituent system requirements belong to different classes; finally, that a balance must be reached for SoS requirements).

V. CONCLUSION

This SLR pursued to understand how Requirements Engineering is planned and executed within the context of System of Systems. More specifically, this paper aimed to search for research challenges that still need to be addressed in the topic. Nineteen studies were included at the end of the SLR and analyzed through reading sheets.

This work is an interesting source for researchers who want to comprehend the challenges or gaps on the field of SoS Requirements and their implications for SoS Engineering. As a future work, we intend to explore one or more of the research challenges encountered in order to contribute to the field of research in Requirements Engineering for System of Systems.

ACKNOWLEDGMENT

CAPES (*Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*) supported this work.

REFERENCES

[1] BIOLCHINI, Jorge et al. Systematic review in software engineering. System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES, v. 679, n. 05, p. 45, 2005.

[2] DAHMANN, J. S. and Baldwin, K. J. (2008). Understanding the current state of us defensesystems of systems and the implications for systems engineering. In Systems Conference, 2008 2nd Annual IEEE, pages 1–7.

[3] DoD-USA (2008). Systems Engineering Guide for Systems of Systems. Washington, DC.

[4] KITCHENHAM, Barbara. Procedures for performing systematic reviews. Keele, UK, Keele University, v. 33, n. 2004, p. 1-26, 2004.

[5] LEWIS, Grace A. et al. Requirements engineering for systems of systems. In: Systems Conference, 2009 3rd Annual IEEE. IEEE, 2009. p. 247-252.

[6] MAIER, Mark W. Architecting principles for systems-of-systems. In: INCOSE International Symposium. 1996. p. 565-573.

[7] BKCASE Editorial Board. 2016. The Guide to the Systems Engineering Body of Knowledge (SEBoK), v. 1.7. R.D. Adcock (EIC). Hoboken, NJ.

[8] WOHLIN, Claes. 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. ACM, 2014. p. 38.

Authenticity Protection in Outsourced Database

Jun Ye
School of Mathematics and Statistics,
Artificial Intelligence Key Laboratory of Sichuan Province,
Sichuan University of Science & Engineering,
Guangxi Key Laboratory of Cryptography
and Information Security,
Zigong, Sichuan, P. R. China
yejun@suse.edu.cn

Zheng Xu *
The Third Research Institute of
the Ministry of Public Security,
Shanghai, China
zhengxu@shu.edu.cn

Yong Ding
Guangxi Key Laboratory of Cryptography
and Information Security,
School of Computer Science
and Information Security,
Guilin University of Electronic Technology,
Guilin, Guangxi, P. R. China
stone_dingy@126.com

Qin Wang
School of Automation
& Information Engineering,
Sichuan University of Science & Engineering,
Zigong, Sichuan, P. R. China
438096157@qq.com

***Abstract*— In this paper, we focus on the security of outsourced database. A verification scheme for outsourced database is proposed based on the verifiable polynomial technique. In this scheme, we consider the encrypted database. The outsourced high degree polynomial will enhance the authenticity of the data. If the cloud server returns fake data, it will be detected by the clients easily.**

Keywords– Verifiable, Polynomial, Outsourced Database

1 Introduction

With the rapid development of Internet and network technology, more and more data will be used. The fast growth of data brings much trouble to people. Cloud computing is powerful, it helps us to solve the big data problem. Cloud computing is the development and application of distributed computing, parallel computing and grid computing. With the help of the powerful cloud server, the client can easily accomplish the complex work easily.

Outsource service allows resource constrained clients to outsource the complex tasks to the powerful cloud server with the manner of pay-per-use. In order to save the own storage space, people will outsource the huge database to a remote cloud server. With the development of cloud computing, outsourcing is becoming an important part in modern business. By utilizing outsourcing, clients can concentrate on their own work and operate their business applications via the Internet, rather than maintaining the substantial hardware, software, and applications.

However, the powerful cloud server is untrusted. When data is outsourced, clients loose the controllability of the outsourced data. It is a hard work to verify the authenticity of the data. The untrusted cloud server may tamper the data. When clients query the data, they will get the fake data. Thus, it is important to study the verifiable outsourced database.

Our Contributions. This paper focus on the correctness of the outsourced data. A new and simple scheme to verify the encrypted data is proposed. In the scheme, the secure oursourcing computation of high degree polynomials is used to help the clients accomplish the verification. Clients can verify the required parts of the outsourced database by checking the *proof* information. The outsourced data is encrypted, and it will not be revealed.

1.1 Related Work

In 1980s, Ben-Or et al. proposed a outsource computation scheme with an honest-but-curious oracle [3, 4]. Then some outsourcing computation

DOI reference number: 10.18293/SEKE2017-027

schemes come out [7, 11]. In 2002, Atallah, Pantazopoulos and Rice [1] proposed secure outsourcing scheme for scientific computing and numerical calculations. However, the verification phase was not considered. In 2008 Benjamin er al. [6] proposed a verifiable outsourcing computation scheme for linear algebraic calculation. And In 2010, Gentry et al. [10] expanded verifiable outsourcing computation to arbitrary function F. However, the efficiency is low. In 2016, Ye et al.[15] proposed a verifiable delegation scheme for polynomials, which improved the efficiency.

There are a lot of work on the verification of outsourced database, such as, [8, 2]. The homomorphic encryption was used in some schemes, however, this reduced the computation efficiency. Then some schemes without homomorphic encryption comes out. Some schemes are based on based on Message Authentication Code [5, 9], and some are based on Merkle hash trees [13, 12]. An index tree for the database is generated by using hash functions, by which the authenticity auditing can be achieved. However, lots of information for verification has to be stored. In 2009, Pang et al. [14] proposed an outsourced database scheme based on the signature chaining technique, in which the computation can be used for verification. In 2013 Catalano and Fiore [9] used the vector commitment to generate a verifiable database with efficient update.

1.2 Organization

The organization of this paper is as follows. Some preliminaries are given in Section 2. The proposed scheme are given in Section 3. Finally, the conclusion is made in Section 4.

2 Preliminaries

2.1 Hash Function

A hash function can take an arbitrary input and output a fixed-size string, which satisfies the following properties.

- It is easy to compute the hash value for any given input.
- It is infeasible to compute an input such that the hash value equals a given hash value.
- It is infeasible to find two different inputs that can get the same hash value.

2.2 Verifiable Outsourced Database

The outsourced database is an example of client-server model. In the outsoutced database model, the service provider is powerful, who has the infrastructure for outsourced databases, and can provide efficient data processing, such as, store, update and query the database.

Verifiable outsourced database allows clients to authenticate database operations. The clients can query the outsourced database, when get the returned results, the clients can verify the correctness of the outsourced data with the help of the proof.

3 The Proposed Scheme

The are three parts in the system, client, cloud server 1 and cloud server 2.

Initialization. The client generates a high degree polynomial, $f(x)$.

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_d x^d$$

where, $a_i \in \mathbb{Z}_p$, $i = 0, 1, \cdots, d$, P is a large prime, and g is a generator.

Then, client selects a prime q, such that $|q| = |p|$, and computes $n = pq$. Client chooses $q_i \in_R \mathbb{Z}_n^*$, and computes

$$r_i = (1+n)^{a_i} q_i^n \mod n^2$$

where, $i = 0, 1, \cdots, d$.

The polynomial is transformed as

$$F(x) = r_0 + r_1 x + r_2 x^2 + \cdots + r_d x^d.$$

Client randomly selects $a, k_0, k_1 \in \mathbb{Z}_p^*$, and computes

$$t_i = g^{k_0 k_1^i} g^{a r_i}$$

where, $i = 0, 1, \cdots, d$.

At last client sends

$$(r_0, r_1, \cdots, r_d)$$

and

$$(t_0, t_1, \cdots, t_d)$$

to cloud server 1.

Ciphertext Generation.The client selects an encryption algorithm $En(\cdot)$ and encrypts the original data x as $\sigma_x = En(x)$, and sends it to the cloud server 1.

When receiving the data σ_x, the cloud server 1 computes

$$\sigma_y = F(\sigma_x) = \prod_{i=0}^{d} r_i^{\sigma_x^i}$$

and

$$T = \prod_{i=0}^{d} t_i^{\sigma_x^i}.$$

Then, cloud server 1 sends σ_y and T to the client.

Client computes

$$Z = \prod_{i=0}^{d} (g^{k_0 k_1^i})^{\sigma_x^i},$$

and

$$y = \frac{(\sigma_y)^{\Phi(n)} - 1}{n} \Phi^{-1}(n).$$

Then, client verifies whether the following equation holds

$$T = Z \cdot g^{ay}.$$

If not, client outputs $\perp$. Otherwise, client computes

$$proof = H(i||\sigma_x||y)$$

where, $H(\cdot)$ is a non-collision hash function.

At last, client sends

$$(i, \sigma_x, proof)$$

to cloud server 2.

Query. The client queries for the data in the position i. Then the cloud server returns the ciphertext

$$C_i = (i, \sigma_{x_i}, H(i||\sigma_{x_i}||y_i))$$

to the client.

Verify. Client sends σ_x to the cloud server 1. Cloud server 1 computes

$$\sigma_{y_i} = F(\sigma_{x_i}) = \prod_{i=0}^{d} r_i^{\sigma_{x_i}^i}$$

and

$$T_i = \prod_{i=0}^{d} t_i^{\sigma_x^i}.$$

Then, cloud server 1 sends σ_{y_i} and T_i to the client.

Client computes Z_i with σ_{x_i},

$$\begin{aligned} Z_i &= \prod_{i=0}^{d} (g^{k_0 k_1^i})^{\sigma_x^i} \\ &= g^{k_0 \frac{1-(k_1\sigma_{x_i})^{n+1}}{1-k_1\sigma_{x_i}}} \end{aligned}$$

and

$$y_i^* = \frac{(\sigma_{y_i})^{\Phi(n)} - 1}{n} \Phi^{-1}(n).$$

And then verifies whether the following equation holds

$$T_i = Z_i g^{a y_i^*}.$$

If not, client outputs $\perp$. Otherwise, client computes

$$proof_i^* = H(i||\sigma_{x_i}||y_i).$$

and verifies

$$proof_i^* \stackrel{?}{=} proof.$$

If the equation holds, the data σ_x is correct, and client can get the original data $x = Dec(\sigma_x)$. Otherwise, client outputs $\perp$.

4 Conclusion

The rapid increasing of data brings much trouble to people. More storage space is needed. Cloud computing gathers a lot of resource together, and provides huge storage space for users. In order to save the local resource, clients often outsource their database to the remote cloud server. And for the security, the outsourced data should be encrypted. As Large amounts of data is outsourced to the cloud server, the users have to keep a little information for verification. In this paper, a new algorithm for outsourced database verification is proposed. The actual data is encrypted, and it will not be revealed. The computation results can be easily verified by the client. The clients can easily check the authenticity of outsourced data.

ACKNOWLEDGMENT

This work was supported by Guangxi Key Laboratory of Cryptography and Information Security (No. GCIS201607); the Fund of Lab of Security Insurance of Cyberspace, Sichuan Province (szjj2016-091); the Talent Project of Sichuan University of Science & Engineering (2017RCL23).

References

[1] M.J. Atallah, K. N. Pantazopoulos, J.R. Rice, and E.H. Spafford. Secure outsourcing of scientific computations. *Advances in Computers*, 54:215–272, 2002.

[2] M. Backes, D. Fiore, and R. M. Reischuk. Verifiable delegation of computation on outsourced data. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 863–874, 2013.

[3] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *Proc. of STAC' 90*, volume 415, pages 37–48. Springer-Verlag, 1990.

[4] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Locally random reductions: Improvements and applications. *Journal of Cryptology*, 10(1):17–36, 1997.

[5] S. Benabbas, R. Gennaro, and Y. Vahlis. Verifiable delegation of computation over large datasets. In *Advances in Cryptology - CRYPTO 2011 - Proceedings of the 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011*, pages 111–131, 2011.

[6] D. Benjamin and M.J. Atallah. Private and cheating-free outsourcing of algebraic computations. In *Sixth Annual Conference on Privacy, Security and Trust, PST 2008, Fredericton, New Brunswick, Canada*, pages 240–245. Springer-Verlag, October 2008.

[7] M. Blum and S. Kannan. Designing programs that check their work. *Journal of the ACM (JACM)*, 42(1):269–291, 1995.

[8] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography, Proceedings of the 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007*, pages 535–554, 2007.

[9] D. Catalano and D. Fiore. Vector commitments and their applications. In *Public-Key Cryptography - PKC 2013 - Proceedings of the 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013*, pages 55–72, 2013.

[10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 465–482, 2010.

[11] O.J. Akomode B. Lees and C. Irgens. Constructing customised models and providing information to support it outsourcing decisions. *Logistics Information Management*, 11(2):114–127, 1998.

[12] E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and integrity in outsourced databases. *TOS*, 2(2):107–138, 2006.

[13] H. Pang, A. Jain, K. Ramamritham, and K. Tan. Verifying completeness of relational query results in data publishing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, pages 407–418, 2005.

[14] H. Pang, J. Zhang, and K. Mouratidis. Scalable verification for outsourced dynamic databases. *PVLDB*, 2(1):802–813, 2009.

[15] J. Ye, H. Zhang, and C. Fu. Verifiable delegation of polynomials. *International Journal of Network Security*, 18(2):283–290, 2016.

Constructing Drug Ingredient Interaction Network to Ensure Medication Security

Zhiren Mao, Pingyi Zhou
State Key Lab of Software Engineering
Computer School, Wuhan University
Wuhan, China
chyanmao@whu.edu.cn
zhou_pinyi@whu.edu.cn

Jiaxiang Zhong, Luojia Jiang
Hubei Shuiguohu High School
Hubei Wuchang Experimental High School
Wuhan, China
zjx001202@gmail.com
zw87652016@gmail.com

Chongzhi Deng, *Jin Liu
Wuhan No.6 High School
Computer School, Wuhan University
Wuhan, China
dczdcz2001@yahoo.com
*Corresponding author
jinliu@whu.edu.cn

***Abstract*—With the fast development of big data, we can do better in modern digital health. The rational use of drugs is main threat of medication security. With big data mining, we analyze the drug instructions to automatically detect the adverse drug interaction in drug combination. The proposed method synthetically employs the NLP and complex network to automatically construct drug ingredient interaction network for mining and inference adverse drug interaction in drug combination. First, many drug instructions are collected by crawler from professional medicine websites. Next, NLP is utilized to automatically extract effective drug ingredient for building drug ingredient library. Then, drug ingredient interactions are extracted from drug instructions. The last, build drug ingredient interaction network. Because the drug ingredient interaction network is kind of complex, the principle of complex network is used to analyze the drug ingredient interaction network. Then we automatically generate a report to make knowledge visualize. The constructed drug ingredient interaction network is verified in experiment. The experiment results indicate that the validity and effectiveness of our proposed method.**

Keywords—Medication security; Big data mining; Complex network; Drug ingredient interaction; Knowledge visualize

I. INTRODUCTION

With the fast development of big data, we can do better in health [1] [2] [3]. According to the World Health Organization (WHO), more than 50% of drugs are prescribed and deployed in unreasonable ways all over the world. Meanwhile, more than 50% patients use drugs in the wrong way. A third of death in patients is attributed to the drug abuse and the irrational use of drug. In American, the irrational use of drugs has been the fourth reason that cause the death and more than 10000 fatal adverse drug events were reported. The problem is more serious in china.

To illustrate this problem, we give a common application scenario. When a person is sick, prescription drugs and the over-the-counter (OTC) drugs can be get easily from online stores (e.g. taobao, JD, etc.). Since potential adverse drug reactions are difficult to recognize by the users themselves. These online platform should generates usage report for its users. For an ordinary user, potential adverse drug reactions are difficult to recognize through the drug instruction. The knowledge from these drug instruction should be extracted and shown to users in an easy way.

In this paper, we mainly focus on how to find irrational drug combination. Drug is made up of auxiliary components and effective components [4]. When two or more drugs are used commonly, effective components may cause changes of pesticide effect which include synergistic effect, antagonism, etc [4] [5]. Reasonable drug interactions can enhance curative effect or reduce the adverse drug reactions. Unreasonable drug interaction in a prescription can reduce curative effect or increase toxicity [6]. So, we construct drug ingredient interaction network to discover and inference irrational drug combination.

This paper firstly utilizes nature language processing (NLP) technology to extract and recognize the drug effective ingredient and the drug ingredient interaction. Next, these drug ingredient interactions are well grouped and marked in type codes. Then build the drug ingredient interaction network. Last, the networks is used to discover and infer adverse drug reactions and the medical report is generated for users.

ADR-Miner approach contains several steps. First, preprocess these crawled drug instructions data into well-structured format. Second step contains two functions: the first function is to recognize and extract drug's effective ingredient and the second function is to recognize and extract drug's ingredient interaction from drug introductions. The third step groups and codes these extracted drug's ingredient interactions. The fourth step constructs the drug ingredient interaction network. The fifth step is to discover and infer adverse drug reactions based on given drug combination and the drug ingredient interaction network. Last, we generate medical report in an easy way for users to understand.

In short, the main contributions of this paper are summarized as follow:

- We formulate a problem that aims to discover the adverse drug reactions from the drug ingredient interaction network for users to avoid irrational drug combination.
- We present an ADR-Miner as a novel analytic framework to tackle this problem, which includes the recognition and extraction of the drug effective ingredient and the drug ingredient interaction and the method to discover and infer adverse drug reaction (ADR).
- We evaluate ADR-Miner based on some irrational drug combination case, in which result shows DI-Miner is effective and promising.

This work is partly supported by the grants of National Natural Science Foundation of China (No. 61572374, U163620068, U1135005).
DOI reference number: 10.18293/SEKE2017-032

The rest of this paper is organized as follow: Section 2 discusses the related work. Section 3 formulates problems. Section 4 covers the technical details of our framework. Section 5 evaluate the performance of our approach. Section 7 concludes.

II. Mining and analyzing drug data

Our work is related to the studies which focus on mining and analyzing drug information in other kinds of information platform (e.g. social media twitter [7], adverse event reporting system (AERS) [8] [9], clinical data [10], professional open data base [11] [12] [13]). However, their methods cannot be directly applied to our problem. In some literatures, there is a lot of work that apply semantic text analysis in different data source [10] [11][14]. At present, all these work are just on drug-to-drug level. This paper firstly utilizes the ingredient-to-ingredient level relations to discover adverse drug reactions.

III. Problem Formulation

In this paper, we formally formulate it as a research problem.

A drug introduction includes an attribute set: the drug name (dn), rating, effective components (ec), known drug interactions (da) and other information. Without loss of generality, in this paper, we choose d = {dn, ec, da}, since we can extract informative information from the common attributes. We mainly focus on the drug name (dn), effective components (ec) and known drug interactions (da). Known drug interactions are described in some sentences sequence $S = \{s_1, s_2, \dots, s_m\}$. Through semantic analysis technology, each sentence s_i describing the known drug interactions is processed into a list which includes elements $adr = < ec_i, ec_j, t_k >$ (i $\neq$ j, t_k is the drug ingredient interaction type). $adr = < ec_i, ec_j, t_k >$ denotes the t_k type drug ingredient interaction between effective component ec_i and ec_j effective component.

In our work, we consider five problems, defined as follow:

Problem 1. How to construct a drug effective ingredient lib $EC = \{ec_1, ec_2, ec_3, \dots, ec_m\}$ from all crawled drug instructions data $D = \{d_1, d_2, d_3, \dots, d_n\}$.

Problem 2. How to construct a drug ingredient interaction list $ADR = \{adr_1, adr_2, adr_3, \dots, adr_l\}$ from all known drug interactions attributes.

Problem 3. How much categories these recognized drug ingredient interaction $ADR = \{adr_1, adr_2, adr_3, \dots, adr_l\}$ can be grouped into.

Problem 4. How to construct the drug ingredient interaction network and whether the drug ingredient interaction network is complex network.

Problem 5. What kind of complex network the drug ingredient interaction network is and how to discover or infer adverse drug reactions based on drug combination$< d_i, d_j >$ (i $\neq$ j) and the drug ingredient interaction network.

IV. Our Framework

Section 1 shows the overview of ADR-Miner framework. In this section, we present each step of our framework in detail.

A. Preprocessing and semantic analyzing

We firstly collect large amount of raw drug instructions from professional medicine website (e.g. http://www.dxy.cn/) by web crawler. Then, we collect more than 32000 drug introductions.

As mentioned in section 3, we choose d = {drug name (dn), effective components (ec), known drug interactions (da)} as research data. The known drug interactions (da) often consist of more than one sentences. We first split da into several sentence $\{s_1, s_2, s_3, \dots, s_h\}$ in this paper.

The raw drug instruction should be converted into sentences. The attribute known drug interactions (*da*) is preprocessed into a list of sentences. For each *da*, we get $da = \{s_1, s_2, s_3, \dots, s_h\}$. In this step, each drug instruction is preprocessed into a tuple $d = \{dn, ec, da = \{s_1, s_2, \dots, s_h\}\}$. If a drug includes more than two effective ingredient, we will not be sure which effective ingredient is involved in known drug interactions. We filter out about 13000 drug introductions without stating its effective ingredient or including more than two or more of the effective ingredient from more than 32000 drug introductions.

B. Extracting

This subsection attempts to address problem 1 and problem 2. The preprocessing step generates a tuple set $D = \{d_1, d_2, \dots, d_n\}$ ($d_i = \{dn, ec, da = \{s_1, s_2, \dots, s_h\}\}$). In this step, our goal is to realize two functions: the first function is to recognize and extract drug's effective ingredient from drug introductions and the second function is recognize and extract drug's ingredient interaction from drug introductions.

Chinese word segmentation system ICTCLAS2015 provided by Zhang Huaping (http://ictclas.nlpir.org/) help us divide a sentence s_i into a word sequence= $\{w_1, w_2, \dots, w_g\}$. Each word w_i in the sequence *W* is marked the part of speech [15]. We firstly extract drug's effective ingredient term to construct a drug's effective ingredient term lib for recognizing the effective ingredient in text written in natural language. For a tuple $d = \{dn, ec, da = \{s_1, s_2, \dots, s_h\}\}$, we can easily extract to construct a term lib $EC = \{ec_1, ec_2, ec_3, \dots, ec_m\}$. Because many drugs contain the same effective ingredient and some drug introductions show they consist more than two effective ingredients that we cannot distinguish which ingredient participates the response of pharmacodynamics described in *da*. So, we extract 2747 kind of effective ingredient from more than 18000 tuple.

The detail effective component lib *EC* is added to the word lib of ICTCLAS system. All the terms in the lib *EC* are set to noun. For each tuple $d = \{dn, ec, da = \{s_1, s_2, \dots, s_h\}\}$, we utilize the ICTCLAS system to divide sentence s_1 into a word sentence. For example, the first sentence of *da* in the drug Benzylpenicillin Sodium for Injection is "Chloramphenicol, erythromycin, tetracycline, sulfa can interfere with the activity of this product". These pharmaceutical ingredients are recognized and converted into four sentence: Chloramphenicol can interfere with the activity of this product, erythromycin can interfere with the activity of this product, tetracycline can interfere with the activity of this product and sulfa can interfere with the activity of this product. So, for each $d = \{dn, ec, da = \{s_1, s_2, \dots, s_h\}\}$, the *da* is updated by the method.

C. Grouping

In this subsection, we attempt to solve problem 3. For a drug induction $d_i = \{dn, ec, da = \{s_1, s_2, \dots, s_h\}\}$, each sentence s_i in d_a is segmented and token by ICTCLAS system. Because of the specification of sentence s_i, we group all these sentences with tool WEKA [16]. By grouping, we summarize 43 kinds of drug ingredient interaction by analyzing the grouping result. All these types are coded. These types can be used as the edge of network in section 4.4 and the template to generate medical report in section 4.6. Fig 1 gives the proportion of each drug reaction types. So, for each sentence s_i in *da*, we generate a tuple $adr = < ec_i, ec_j, t_k >$ ($i \neq j$, t_k is the drug ingredient interaction type). In the tuple *adr*, ec_i is the *ec* attribute of d_i and ec_i is recognized from sentence s_i. By processing, we recognizes more than 11000 drug ingredient interaction $ADR = \{adr_1, adr_2, adr_3, \dots, adr_l\}$.

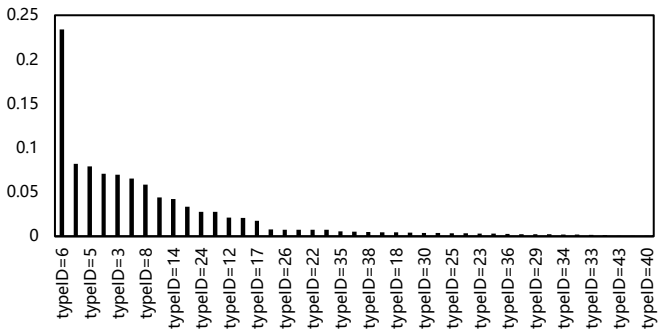


Fig. 1. The proportion of each adverse drug action types

D. Constructing network

This subsection attempts to address problem 4. We get more than 2700 effective ingredients $EC = \{ec_1, ec_2, \dots, ec_m\}$ in section 4.2 and recognized more than 11000 drug ingredient interaction $ADR = \{adr_1, adr_2, \dots, adr_l\}$ in section 4.3. This section utilize these information to construct ingredient-to-ingredient level drug ingredient interaction network.

To describe the relations among effective ingredients, we defined a directed network *DIN* = (*V*, *E*). Each effective ingredient ec_i in *EC* is regarded as a node *v* in network *DIN*. So, *EC* can be regarded as the node set *V* of *DIN*. For each $adr = < ec_i, ec_j, t_k >$, *adr* is converted to an edge *e* in which ec_i is the tail and ec_j is the head. t_k in *adr* denotes the type of the edge *e*. A matrix *A* is used to denote the network *DIN*. If there is a drug ingredient interaction between node v_i and v_j, the element a_{ij} of the matrix *A* denotes the type t_k. If not, a_{ij} is equal to 0.

$$A = \begin{bmatrix} a_{1,1} & \dots & a_{1,N} \\ \dots & \dots & \dots \\ a_{N,1} & \dots & a_{N,N} \end{bmatrix} \quad (1)$$

Fig.2 shows the drug ingredient interaction network which includes more than 2700 nodes and more than 11000 edges. With complex networks theory we identify what kind of complex network the drug ingredient interaction network is and analyze some feature of the network. Some important index to evaluate a network are graph density [17], degree distribution [18], average path length [19], clustering coefficient [20] and etc.

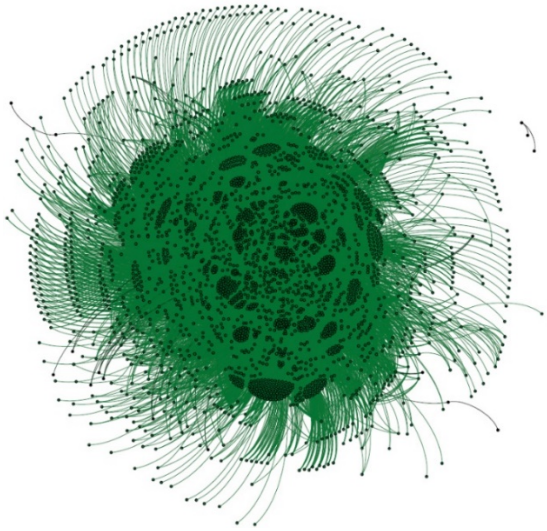


Fig. 2. The drug ingredient interaction network

Graph density *GD* is used to evaluate the closeness among nodes. *GD* is defined in equation (2). For a complete graph, the graph density of the complete graph is 1. By computing, the graph density *GD* of the network is 0.014. So, the drug ingredient interaction network is a sparse one. Because the known drug reaction came from report and drug test, there are a lot of unknown drug reactions in more than 98% the remaining space.

$$GD = \frac{|E|}{2 * N(N-1)} \quad (2)$$

The degree of a node *v* in *DIN* is the number of the edges incident to the node. The degree of a node *v* is denoted deg(*v*) or d(*v*). The in-degree of a node *v* in *DIN* is the number of the edges in which the node is the head. The in-degree of a node is denoted $d^-(v)$. The out-degree of a node *v* is the tail, which is denoted $d^+(v)$. In the network, the distribution function *P*(*k*) is used to denote the number of nodes whose degree is equal to *k*. *P*(*k*) is defined in equation (3). Similarity, the in-distribution function $P^-(k)$ and the out-distribution function $P^+(k)$ is defined in equation (4) and (5).

$$P(k) = \sum_{i=1}^{n} \prod(\deg(v_i) == k) \quad (3)$$

$$P^-(k) = \sum_{i=1}^{n} \prod(\mathrm{d}^-(v_i) == k) \quad (4)$$

$$P^+(k) = \sum_{i=1}^{n} \prod(\mathrm{d}^+(v_i) == k) \quad (5)$$

$\prod(x)$ is an indicator function. When the x is true, the value of the indicator function is 1. When false, it is 0.

Generally speaking, the degree of a node *v* measures the degree of important of *v* in *DIN*. The average degree <*k*> is generated by a simple formula (6). By computing, the average degree <*k*> is 85.121.

$$< k >= \frac{1}{N} \sum_{i=1}^{N} dev(v_i) \quad (6)$$

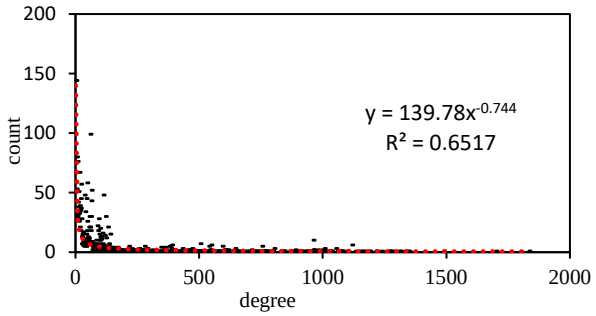


Fig. 3. The degree distribution of the network

Fig.3 shows the degree distribution of the network *DIN*. Fig.4 and fig. 5 show the in-degree and out-degree distribution of the network. We found the in-degree distribution contributing many low-degree value and out-degree distribution contributing many high-degree value. Because the more large degree means the more important role in the network. In the drug ingredient interaction network, most of the nodes have only a few links

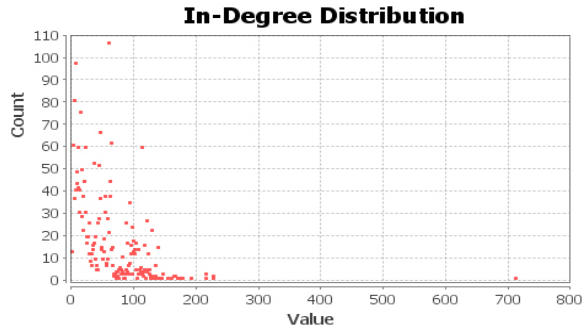


Fig. 4. The in-degree distribution of the network

while a few nodes have a large number of links to other nodes. By curve fitting, we get the distribution function $P(k) = 139.78^{-0.744}$. The multiple correlation coefficients R^2 which is an index to evaluate the degree of linear correlation between the variables is 0.6517. Because the degree distribution is power law distribution, the drug ingredient interaction network is a scale-free network [21][22].

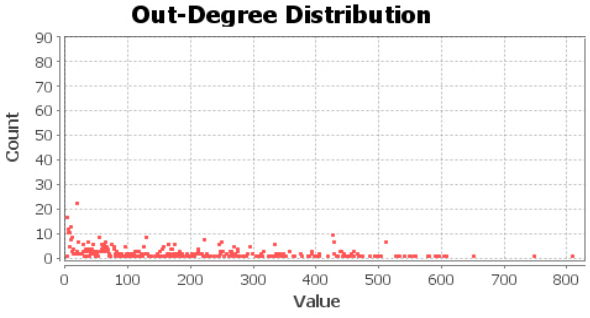


Fig. 5. The out-degree distribution of the network

Centrality is an important index to evaluate the relative importance of nodes of a network. There are many method to measure centrality. This paper utilizes eigenvector centrality. For each node v_i, a relative score x_i is assigned to the node. In the relative score of a node v_i, these links to high relative score nodes have larger contribution than these links to low relative score nodes. The formula of node relative score x_i is given as follow:

$$x_i = \frac{1}{\lambda}\sum_{j}^{N}\prod(a_{ij})x_j \qquad (7).$$

In the formula (7), N is the number of nodes and λ is a constant. All these nodes relative score in the network is regarded as a vectory $x = \{x_i, x_2, \dots, x_N\}$. So, the formula (7) is converted into formula (8). In the formula (8), each characteristic vector solution corresponds to different characteristic constant. Each node v_i in the network is assigned a relative score x_i by solving the formula (8). Fig.6 shows the

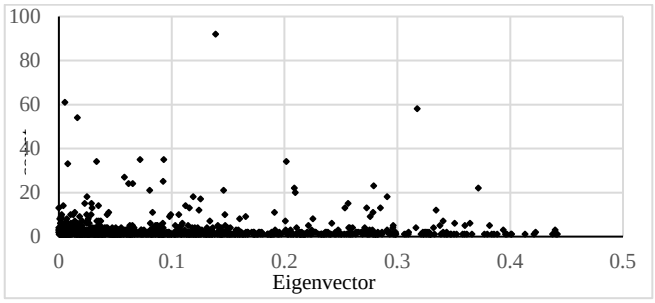


Fig. 6. The eigenvector centrality distribution of the network

eigenvector centrality distruction of the netwok. The the eigenvector centrality distruction is shown in equation (9).

$$Ax = \Phi x \ (\mathrm{x} = \{x_1, x_2, \dots, x_N\}, \Phi = \{\lambda_1, \lambda_2, \dots, \lambda_N\}) \qquad (8)$$

$$EP(k) = \sum_{i=0}^{N}\prod(x_i == k) \qquad (9)$$

In the network *DIN*, the distance between v_i and v_j is defined as the shortest path length of the two nodes and denoted as d_{ij}. If the node v_i and v_j is disconnected, $dis_{ij} = \infty$. The maximum distance of any two nodes in a network is the diameter of the network. The network diameter is get by the formula:

$$Dia = \max_{1\le i<j\le N} dis_{ij} \qquad (10)$$

The diameter of the network *DIN* is 6. The average distance of any two nodes in a network is the average path length of a network. The formal definition is given as equation (11). By computing, the average path length of the network *DIN* is 2.46. So, the network *DIN* has some characteristics of small-world network [23]. For a small-world network, the network not only has short average path distance, but also has large clustering coefficient.

$$L = \frac{1}{C_N^2}\sum_{1\le i<j\le N} dis_{ij} \qquad (11)$$

For a node v_i in a network G, there are $\deg(v_i)$ edges connecting with other $\deg(v_i)$ nodes. These $\deg(v_i)$ nodes is called the neighbor of the node v_i. There are at most $C^2_{\deg(v_i)}$ edges between these $\deg(v_i)$ nodes. The clustering coefficient C_i of the node v_i is the ratio of the actual edges AE_i between these $\deg(v_i)$ nodes to $C^2_{\deg(v_i)}$. Formally,

$$C_i = \frac{AE_i}{C^2_{dev(v_i)}} \qquad (12)$$

The clustering coefficient of the network is the average clustering coefficient of all node in the network. Formally,

$$C = \frac{1}{N}\sum_{i=1}^{N} C_i \tag{13}$$

By computing, the average clustering coefficient of the network is 0.201.

Through the analysis of the above, we evaluate some characteristics of the network *DIN*. We find that the network *DIN* not only is a scale-free network but also a small-world network.

E. Finding and inferencing

We construct the drug ingredient interaction network above. In this subsection, we attempt to solve problem 5. So, we give a method to automatically discover and infer adverse drug reactions based on given drug combination and the drug ingredient interaction network.

When a user purchase some drug in online store (e.g. taobao, JD, etc.). Because of the limited cognitive ability, some potential adverse drug reactions are very difficult to find by the user itself. A drug in an online store contains the drug's introduction. We can recognize the drug effective ingredients from the drug introduction. Based on this principle, we can recognize all these effective ingredients of these drugs which are in user shopping cart. These effective ingredients can have some adverse drug reactions. Utilize the constructed drug ingredient interaction network, these potential adverse drug reactions can be detected.

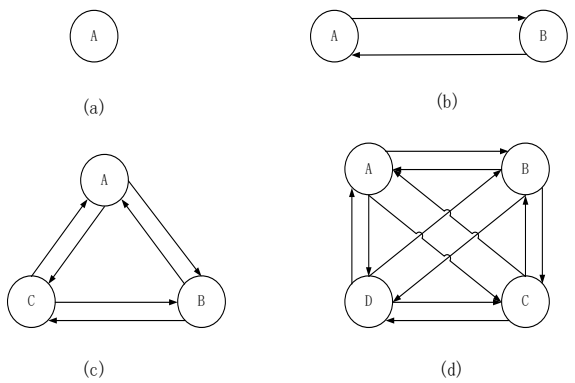


Fig. 7. The candidate set of potential adverse drug reactions

Fig. 7 shows the generating candidates principle. When the effective ingredient set *Eu* recognized form *Du* only contains a kind of effective ingredient, the fig.7 (a) shows the circumstance. In this circumstance, there is no potential adverse drug reactions. When the effective ingredient set *Eu* contains two kinds of effective ingredient, the fig.7 (b) shows the circumstance. In this circumstance, we first generate a candidate set $= \{A \rightarrow B, B \rightarrow A\}$. In the candidate set *H*, candidate element h_i is a binary sequence. For each candidate element h_i in the candidate set *H*, if h_i can be found in the drug ingredient interaction network, the candidate element h_i and corresponding drug in-gredient interaction type t_k which can be used to generate re-port are add to the result set *R*. So, the element of *R* is two-tuple $r =< h_i, t_k >$. Fig.7. (c) and fig.7. (b) Separately shows circumstance of |*Eu*| = 3 and |*Eu*| = 4. If the effective ingredient set |*Eu*| = m, the number of the element of the candidate set *H* is equal to m(m-1)*2.

Given the ingredients term lib *EC*, the drug set *Du* and the drug ingredient interaction network *DIN*, a method to discover and infer adverse drug reactions is shown in algorithm 1.

Algorithm 1. Find and inference algorithm for detecting adverse drug reactions

Input:

EC : the ingredients term lib.

Du : the drug combination.

DIN : the drug ingredient interaction network.

Output:

R : these discovered adverse drug reactions.

1. For the drug set $Du = \{d_1,...,d_n\}$, we utilize the ingredients term lib *EC* to recognize all these effective ingredients $Eu = \{e_1,...,e_m\}$ form the drug set $Du = \{d_1,...,d_n\}$.
2. For these effective ingredients $Eu = \{e_1,...,e_m\}$, we generate the candidate set $H = \{h_1, h_2,..., h_{m(m-1)*2}\}$.
3. For each element h_i in the candidate set H, we utilize the drug ingredient interaction network *G* to detect whether the element h_i is a kind of adverse drug reactions. If the element h_i is a kind of adverse drug reactions, a tuple $< h_i, t_k >$ is add to the result set *R*.

F. Generating interpretation and knowledge visualization

For a given drug combination *Du*, we utilize algorithm 1 to get a result set *R*. For the result set *R*, if the *R* is an empty set. It means that no adverse drug reactions is found in the given drug combination *Du*. If the *R* is not an empty set, for each element $r =< h_i, t_k >$ in the set *R*, we utilize the corresponding drug ingredient interaction type t_k to illustrate the relation in $h_i =< e \rightarrow e >$. A report is generated for user to illustrate the result.

V. EVALUATION

To evaluate if ADR-Miner can really help users to recognize adverse drug reactions, we conduct the evaluation. We first collect reported medical cases to form the test set. Then we introduce the metric used in our evaluation. Last, based on the test case and the metric, we evaluate the performance.

A. Test set

In this subsection, we collect some common adverse drug reactions to form test set. Without loss of generality, a large number of reported adverse drug reaction cases are not selected as the test sample. Some common adverse drug reactions which have been recorded in the common drug incompatibility table are selected. We select 260 adverse drug reactions from the common drug incompatibility table which have been verified.

B. Big data Performance analysis

In this section, we introduce the metric used in our evaluation. These metrics include precision, recall and F-measure which are defined as follow:

$$precision = \frac{TP}{TP + FP} \tag{14}$$

$$recall = \frac{TP}{TP + FN} \tag{15}$$

$$F - measure = \frac{2 * precision * recall}{precision + recall} \tag{16}$$

Where *TP*, *FP*, *FN* separately denote the number of the true positive, false positive, and false negative.

C. Evaluation of framework performance

In this section, we utilize the test set constructed above to evaluate the performance of the framework. For all samples in the test set, every sample is the true adverse drug interaction sample. If a sample is recognized by the framework, it is a true positive (TP). If a sample is not recognized by the framework, it is a false negative (FN). Utilizing the metrics, the precision is 1, the recall is 0.51 and the F-measure is 0.68.

VI. Conclusion

The adverse drug interaction is a serious threat for medicine security. This paper proposed DI-Miner framework for ordinary users to avoid adverse drug interactions. The approach employs NLP, big data mining, knowledge visualization, complex network technology to create a drug ingredient interaction network. The network is used to tell whether a drug combination has adverse drug interactions. The proposed method can not only eliminate the gap between the human's limited cognitive ability and the medicine knowledge but also make health service more intelligent. The experiment and analysis proved the validity and effectiveness of our approach.

As we all know, it is the first time that an ingredient-level method has been used to detect adverse drug interaction. And we are now planning for further expansion. First, since we haven't utilized all the information of drug introductions, the method only recognize generalization adverse drug interaction and the patients' physical information is not considered in our method. Second, a detecting report will be generated for users, the report does divide the danger level of recognized adverse drug reactions. We will explore an automatic method for dividing the danger level in our future work [24][25].

Acknowledgment

Jin Liu is the corresponding author. We would like to thank Pingyi Zhou for his constructive comments which help a lot. This work is partly supported by National Natural Science Foundation of China (NSFC) (grant No. 61572374, U163620068, U1135005), the Academic Team Building Plan from Wuhan University and National Science Foundation (NSF) (grant No. DGE-1522883).

References

[1] Xu Z, Yan Z, Mei L, et al. The Big Data Analysis of the Next Generation Video Surveillance System for Public Security[C]//Workshop on E-Business. Springer International Publishing, 2015: 171-175.

[2] Luo X, Liu Y, Li Q. Big data - related technologies and applications[J]. Concurrency and Computation: Practice and Experience, 2016, 28(15): 4036-4037.

[3] Xu Z, Mei L, Hu C, et al. The big data analytics and applications of the surveillance system using video structured description technology[J]. Cluster Computing, 2016, 19(3): 1283-1292.

[4] Brown S D, Nativo P, Smith J A, et al. Gold nanoparticles for the improved anticancer drug delivery of the active component of oxaliplatin. Journal of the American Chemical Society, 2010, 132(13): 4678-4684.

[5] Azizi M, Ménard J, Bissery A, et al. Pharmacologic demonstration of the synergistic effects of a combination of the renin inhibitor aliskiren and the AT1 receptor antagonist valsartan on the angiotensin II–renin feedback interruption. Journal of the American Society of Nephrology, 2004, 15(12): 3126-3133.

[6] Drug-drug interactions. CRC Press, 2008.

[7] Bian J, Topaloglu U, Yu F. Towards large-scale twitter mining for drug-related adverse events. Proceedings of the 2012 international workshop on Smart health and wellbeing. ACM, 2012: 25-32.

[8] Harpaz R, DuMouchel W, Shah N H, et al. Novel Data-Mining Methodologies for Adverse Drug Event Discovery and Analysis. Clinical Pharmacology & Therapeutics, 2012, 91(6): 1010-1021.

[9] Harpaz R, Chase H S, Friedman C. Mining multi-item drug adverse effect associations in spontaneous reporting systems. BMC bioinformatics, 2010, 11(9): 1.

[10] Pathak J, Kiefer R C, Chute C G. Using linked data for mining drug-drug interactions in electronic health records. Studies in health technology and informatics, 2013, 192: 682.

[11] Lindquist M, Ståhl M, Bate A, et al. A retrospective evaluation of a data mining approach to aid finding new adverse drug reaction signals in the WHO international database. Drug Safety, 2000, 23(6): 533-542.

[12] Wishart D S, Knox C, Guo A C, et al. DrugBank: a comprehensive resource for in silico drug discovery and exploration. Nucleic acids research, 2006, 34(suppl 1): D668-D672.

[13] Wong C M, Ko Y, Chan A. Clinically significant drug–drug interactions between oral anticancer agents and nonanticancer agents: profiling and comparison of two drug compendia. Annals of Pharmacotherapy, 2008, 42(12): 1737-1748.

[14] Xie F, Xu Z. Semantic based annotation for surveillance big data using domain knowledge[J]. International Journal of Cognitive Informatics and Natural Intelligence (IJCINI), 2015, 9(1): 16-29.

[15] Xia F. The part-of-speech tagging guidelines for the Penn Chinese Treebank (3.0). 2000.

[16] Zhang K, Zan H, Chai Y, et al. Survey of the Chinese Function Word Usage Knowledge Base. Journal of Chinese Information Processing, 2015.

[17] Scott J. Social network analysis. Sage, 2012.

[18] Dorogovtsev S N, Mendes J F F, Samukhin A N. Size-dependent degree distribution of a scale-free growing network. Physical Review E, 2001, 63(6): 062101.

[19] Fronczak A, Fronczak P, Hołyst J A. Average path length in random networks. Physical Review E, 2004, 70(5): 056110.

[20] Zhou T, Yan G, Wang B H. Maximal planar networks with large clustering coefficient and power-law degree distribution. Physical Review E, 2005, 71(4): 046141.

[21] Holme P, Kim B J. Growing scale-free networks with tunable clustering. Physical review E, 2002, 65(2): 026107.

[22] Pastor-Satorras R, Vespignani A. Epidemic spreading in scale-free networks. Physical review letters, 2001, 86(14): 3200.

[23] Newman M E J, Watts D J. Renormalization group analysis of the small-world network model. Physics Letters A, 1999, 263(4): 341-346.

[24] Liu Z, Wei C, Ma Y, et al. UCOR: an unequally clustering-based hierarchical opportunistic routing protocol for WSNs[C]//International Conference on Wireless Algorithms, Systems, and Applications. Springer Berlin Heidelberg, 2013: 175-185.

[25] Liu Z, Niu X, Lin X, et al. A Task-Centric Cooperative Sensing Scheme for Mobile Crowdsourcing Systems[J]. Sensors, 2016, 16(5): 746.

Is the Number of Faults Helpful for Cross-Company Defect Prediction?

Yiyang Jing[1], Jiansheng Zhang[2], *Jin Liu[1]
[1]*State Key Lab. of Software Engineering*, Computer School, Wuhan University, Wuhan, China
[2]*School of Computer Science and Information Engineering*, HuBei University, Wuhan, China
*Corresponding author
*Corresponding author email: jinliu@whu.edu.cn

***Abstract*—In the field of Cross-company Defect Prediction (CCDP), how to deal with the data to make it more accurately predict the cross-company software defects is the focus problem we need to consider. Now a mainstream idea is to determine the weight of the training data based on the similarity between the training data and the test set, and then build the model on the basis of these weighted data. However, sometimes, when we deal with some problems with imbalance class, directly using the weight calculated above may lead to errors. Because a large number of non-defective instance's weight accumulation will lead to the defective instance's weight has a little impact on the final result. This is why we need to consider the addition of the number of defects. Considering the number of defects will effectively eliminate the impact of a large number of non-defective instance's weight accumulation. Therefore, we propose a Transfer-learning Naïve Bayes model considering the number of defective information(NTNB). The method consists of two major stages: weight the data and build the prediction model. In the stage of weighting the data, we not only consider the similarity between the data but also consider the number of defects to get the final weights for data. And we conducted a set of comparative experiments on six open cross-company datasets. The results show that considering the number of defects information can effectively avoid some defective instance is misjudged as non-defective instance, and improve the accuracy of prediction in some unbalanced problems.**

Keywords—software defect prediction;cross-company defect prediction; transfer learning; NTNB

I. INTRODUCTION

Software defect prediction is one of the most important software quality assurance techniques. It aims to detect the defect proneness of new software modules via learning from defect data. So far, many efficient software defect prediction approaches [1-2] have been proposed, but they are usually confined to within company defect prediction (WCDP). WCDP works well if sufficient data is available to train a defect prediction model. However, it is difficult for a new company to perform WCDP if there is limited historical data. Cross-company defect prediction (CCDP) is a practical approach to solve the problem. It trains a prediction model by exploiting one or multiple projects of a source company and then applies the model to target company [3].

Recent year, there have sprung up a large number of excellent articles to solve the cross-company defect prediction problem. For example, Turhan et al. use a Nearest Neighbor Filter model (NN-filter) to select the similar data from source data as training data for more accurate forecasts [4]. Be different from this, Ying Ma et al. select all the source data as training data but add weights to them to build a Transfer Naive Bayes model(TNB)[5]. However, all of these models are not considered the number of faults whether has an impact on the forecast results. That is to say, the information about the number of faults has not been fully utilized.

However, when calculating the weight between training instances and test instances, the number of faults information has an important additive effect on it. For example, we tend to give a higher weight to an instance which has fewer bugs, and these higher weights allow these instances play a greater role in predicting the presence or absence of defects in the test instance. For instance, when calculating the similarity between instances, we get the same degree of similarity between several training instances and the test instance, so which of them is more convincing in determining whether the test instance is defective? In this time, we need to take the number of faults information into consider.

Therefore, this paper firstly gives the corresponding initial weights to the cross-project training instances through the Transfer Learning [6]. Then give the final weight combines with number of faults information and calculate the prior probabilities of the presence or absence of defects. Finally build the forecast model (NTNB) on them. We select six available and commonly used software project datasets and choose five of them as training data, the remaining one as the testing data to do the experiments [7]. The experimental results show that the NTNB model which considering the number of faults is superior than the traditional prediction models.

The structure of this article is as follows, Section II is the related work in this area; Section III introduces the NTNB approach for CCDP. Section IV shows the experiment setup and experiment results. Section V addresses the conclusion and points out the future work.

II. RELATED WORK

In recent years, Machine Learning is more and more widely applied in various fields [8-10], In the field of defect prediction, the machine learning methods are also used a lot. For example, Xiao Yu propose a novel probabilistic graphical model called Bayesian Network based Program Dependence Graph (BNPDG) to locate the defect in the software. In addition, decision trees,

10.18293/SEKE2017-034

neural networks and other methods have also been widely used to predict the defect of software [23-24].

However, with the evolution of the algorithm, it has been found that it is more difficult to improve the prediction accuracy only by improving the algorithm [5]. So people began to improve the prediction accuracy by using more appropriate training data. For example, by removing the data which is poorly related to the company, to mitigate the impact of irrelevant data on forecasting [15].

In the process of cross-company forecasting(CCDP), The choice of training data becomes more important. Because the cross-company defect prediction often means that in the prediction process, the training data and test data have different feature distribution and prior knowledge. This requires us to process the original data to eliminate this priori difference. In order to achieve this purpose, Turhan et al. use a Nearest Neighbor Filter model (NN-filter) to select the similar data from source data as training data for more accurate forecasts [4], Ying Ma et al. select all the source data as training data but add weights to them to build a Transfer Naive Bayes model(TNB) [5].

In addition to selecting the appropriate training data, people also thought that by selecting the valuable characteristics to improve the accuracy of prediction. Zhou Xu proposed a Maximal Information Coefficient with Hierarchical Agglomerative Clustering (MICHAC) method, first of all, sorting the attributes according to the amount of information of them, and then remove the redundant attributes through the hierarchical clustering to get the most valuable attributes [14] [16].

III. METHODOLOGY

In this section, we present our NTNB approach for CCDP. The method consists of two major stages: weight the data and build the prediction model. In the stage of weighting the data, we not only consider the similarity between the data but also consider the number of defects to get the final weights for data. In the stage of building the prediction model, first of all, we calculate the prior probabilities and conditional probabilities based on the previously calculated weights, and then use the principles of the Naïve Bayes classifier to predict the label of the test instance.

A. The Naive Bayes Model

The reason why do we use the Naive Bayes as our forecasting model is that when we encounter such a problem which need to consider all attribute information to get the probability of the result, many algorithms may ignore some weak features, but the Naive Bayes model will use all available information to correct the forecast results. It's important for the defect prediction field which need to consider a large number of attributes information. Although many attributes may have little impact on the result separately, but the combination of them will make a great influence.

The Naïve Bayes' thought is to calculate the probability that an instance belongs to each category under given conditions, and which category's probability is higher, which category we think it belongs to.

For example, there is an instance x, and the category set is $C = \{c_1, c_2\}$. If the probability of x belongs to the category c_1 is larger than it belongs to c_2, then we regard the instance x's category as c_1, otherwise as c_2.

$$P(c_1|x) > P(c_2|x) \rightarrow x \in c_1 \quad (1)$$

$$P(c_1|x) < P(c_2|x) \rightarrow x \in c_2 \quad (2)$$

And $P(c_i \mid x) = \frac{P(c_i)*P(x \mid c_i)}{P(x)}$ .The $P(c_i)$ is the prior probability of class c_i, The $\frac{P(x \mid c_i)}{P(x)}$ is an adjustment factor to adjust the value of the posteriori probability, in order to make it more close to the true probability.

B. The Weight Of Training Data

This part we intend to calculate the weight of the training data by considering the number of faults and the distance between training instances and test instances.

First, we measure the distance between instances by Transfer Learning [6]. The goal of Transfer Learning is to move the knowledge learned from one environment to deal with the problem in a new environment. It's particularly suitable for the cross-company defect prediction, because that the cross-company defect prediction is just using the cross-company code defect information to predict the code deficiencies for different companies. Then how do we determine the weights of training instances in the migration process? An important principle is: the higher the similarity between the training instance and the testing instance, the higher the weight.

Then we first calculate the similarity between instances and calculate the weights based on similarity.

According to TNB [5], the similarity between instances is measured as follows: 1) we need to find the maximum and minimum values of each attribute in the test set and store them into arrays.2) Next we judge whether each attribute of each training instance is within the maximum and minimum range of the corresponding attribute in the test set. If yes, the corresponding instance's support factor will plus one. For example, giving three training instances: x_1= (1, 2, 2, 'false'), x_2= (2, 1, 3, 'false'), x_3= (2, 2, 4, 'true'|3), where the bit after 'true' is the number of defects, and one training instance: y_1= (2, 2, 3). Then Max = (2, 2, 3), Min = (2, 2, 3). So for the first instance = (1, 2, 2, 'false'), the support factor $s_1 = 1$; Similarly, $s_2 = 2$, $s_3 = 2$.

Next, we will calculate the weight of each training instance by the above support factors. Here, according to L. Peng's [17] paper, we use the gravitational formula to simulate the gravitational force between the data, that is the weight.

$$w_i = G\frac{m_1 m_2}{(r)^2} = \frac{kmMs_iM}{(k - s_i + r)^2} \propto \frac{s_i}{(k - s_i + r)^2} \quad (3)$$

In Eq.(3), w_i is the weight of each training instance; G is the universal gravitational constant, m_1 and m_2 is the mass of two objects. r is the distance between the two objects; k is the number of attributes, m is the number of test cases, M is the mass of each attribute, so kmM is the mass of all test cases; s_i is the support factor of each training instance, so s_iM is the mass of each training instance; Similarity, $(k-s_i+1)$ is the distance between each training instance and the test set. In this process, we can remove some fixed constants, and thus get the right part

of the Eq.(3). In the end, we can get the final weight w by accumulate each training instances' weight w_i.

But that's not our final weights. Sometimes, when we deal with some problems with imbalance class, directly using the value calculated above may lead to errors. Because a large number of non-defective instance's weight accumulation will lead to the defective instance's weight has a little impact on the final result. For example, for training instances x_1= (1, 2, 2, 'false'), x_2= (2, 1, 3, 'false'), x_3= (2, 2, 4, 'true'|3), and the test instance: y_1= (2, 2, 3). According to Eq.(3), we can calculate the weight of each training instance, w_1 = 1/9, w_2 =1/2, w_3 = 1/2. If we predict the presence of defects directly according to these weights, then the test instance is likely to be judged as having no errors due to the accumulation of the non-defective instance's weight. However, based on the similarity between the test instance and the defective training instance, this test instance is still likely to be defective. So, calculating weights without considering the number of defects can cause some defective instance is mistaken for non-defective instance, which will reduce the prediction accuracy.

Therefore, we consider the number of defects information in calculating the weight of each instance. That is to say we add the number of defects on the basis of Eq.(3).

$$w_i = \frac{s_i n_i}{(k - s_i + 1)^2} \tag{4}$$

n_i is the number of defects for the *i*-th training instance.

Next we will describe how to predict the presence of defects on the basis of this weight.

C. NTNB Approach

The general idea of our algorithm is to use the Naïve Bayes as our forecasting model on the weighted data. According to Eq.(1) and Eq.(2), in order to calculate the posterior probability $P(c_i|\, x)$, we need to calculate $P(c_i)$, $P(x \mid c_i)$ and $P(x)$. To better explanation, we define the indicative function $f(x, y)$: if x = y, $f(x, y) = 1$, otherwise, $f(x, y) = 0$.

According to [22], The prior probabilities $P(c)$ can be expressed as:

$$P(c) = \frac{\sum_{i=1}^{n} f(c_i,c) w_i + 1}{\sum_{i=1}^{n} w_i + n_c} \tag{5}$$

n is the total number of training instances, $f(c_i, c)$ is the indication function described above, c_i is the category of the i-th training instance, c is the label of the attribute that we want to calculate the prior probability, w_i is the weight of the *i*-th training instance, and n_C is the total number of categories.

Next, we will calculate the conditional probability of the *j*-th attribute a_j in the training instance x_i according to the formula in [18].

$$P(a_j|c) = \frac{\sum_{i=1}^{n} f(a_{ij},a_j) f(c_i,c) w_i + 1}{\sum_{i=1}^{n} f(c_i,c) w_i + n_j} \tag{6}$$

n is the total number of training instances, a_{ij} is the value of *j*-th attribute in *i*-th training instance, c_i is the category of the *i*-th training instance, c is the label of the attribute that we want to calculate the conditional probability, w_i is the weight of the *i*-th training instance, and n_j is the number of different values for attribute a_j in the training set.

Suppose that x is an instance. Then:

$$P(x) = \sum_{i=1}^{n_c} P(c_i) \prod_{j=1}^{k} P(a_j \mid c_i) \tag{7}$$

n_C is the total number of categories, k is the number of attributes in instance x, $P(ci)$ is introduced in Eq.(5), $P(aj \mid ci)$ is introduced in Eq.(6).

So we can predict the x's category by:

$$P(c_i \mid x) = \frac{P(c_i) \prod_{j=1}^{k} P(a_j| \, c_i)}{P(x)} \tag{8}$$

The $P(c_i| \, x)$ is the probability that instance x is predicted as class c, $P(c_i)$ is introduced in Eq.(5), k is the number of attributes in instance x, $P(aj \mid ci)$ is introduced in Eq.(6), $P(x$ is introduced in Eq.(7).

If $P(c_i \mid x) > P(c_i \mid x), 1 \ll j \ll nc, j \neq i$,then, $x \in c_i$. Otherwise, $x \in c_j$. where n_c is the total number of categories.

Next, we will classify the below examples from considering the number of defects and not considering it.

For training instances x_1= (1, 2, 2, 'false'), x_2= (2, 1, 3, 'false'), x_3= (2, 2, 4, 'true'|3), and the test instance: y_1= (2, 2, 3). Then n = 3, n_C = 2, k = 3.

1) Considering The Number Of Defects

① According to Eq.(4), we can calculate the weight of each training instance, w_1 = 1/9, w_2 =1/2, w_3 = 3/2.

② According to Eq.(5)

$$P('true') = \frac{(w_3 * 1) + 1}{(w_1 * 1 + w_2 * 1 + w_3 * 1) + 2} = 0.608$$

$$P('false') = \frac{(w_1 * 1 + w_2 * 1) + 1}{(w_1 * 1 + w_2 * 1 + w_3 * 1) + 2} = 0.392$$

③ According to Eq.(6), n_1=2 ,n_2=2, n_3= 3.

$$P(a1 = 2|'true') = \frac{(w_3 * 1) + 1}{(w_3 * 1) + 2} = 0.714$$

$$P(a1 = 2|'false') = \frac{(w_2 * 1) + 1}{(w_1 * 1 + w_2 * 1) + 2} = 0.574$$

$$P(a2 = 2|'true') = \frac{(w_3 * 1) + 1}{(w_3 * 1) + 2} = 0.714$$

$$P(a2 = 2|'false) = \frac{(w_1 * 1) + 1}{(w_1 * 1 + w_2 * 1) + 2} = 0.426$$

$$P(a3 = 3|'true') = \frac{1}{(w_3 * 1) + 3} = 0.222$$

$$P(a3 = 3|'false') = \frac{(w_2 * 1) + 1}{(w_1 * 1 + w_2 * 1) + 3} = 0.415$$

④ According to Eq.(7),

$$P(y_1) = P('true') \prod_{j=1}^{3} P(a_j \mid 'true') + P('false') \prod_{j=1}^{3} P(a_j \mid 'false')$$

$$= 0.0688 + 0.0398 = 0.1086$$

⑤ According to Eq.(8)

$$P('true'|y_1) = \frac{P('true')\prod_{j=1}^{3}P(a_j \mid 'true')}{P(y_1)} = 0.6335$$

$$P('false'|y_1) = \frac{P('false')\prod_{j=1}^{3}P(a_j \mid 'false')}{P(y_1)} = 0.3665$$

Obviously, $P('true'|y_1) > P('false'|y_1)$, So the test instance y_1's category is true.

2) Not Considering The Number Of Defects

① According to Eq.(4), we can calculate the weight of each training instance, $w_1 = 1/9$, $w_2 = 1/2$, $w_3 = 1/2$.

② According to Eq.(5)

$$P('true') = \frac{(w_3 * 1) + 1}{(w_1 * 1 + w_2 * 1 + w_3 * 1) + 2} = 0.482$$

$$P('false') = \frac{(w_1 * 1 + w_2 * 1) + 1}{(w_1 * 1 + w_2 * 1 + w_3 * 1) + 2} = 0.518$$

③ According to Eq.(6), $n_1=2$,$n_2=2$, $n_3=3$.

$$P(a1 = 2|'true') = \frac{(w_3 * 1) + 1}{(w_3 * 1) + 2} = 0.6$$

$$P(a1 = 2|'false') = \frac{(w_2 * 1) + 1}{(w_1 * 1 + w_2 * 1) + 2} = 0.574$$

$$P(a2 = 2|'true') = \frac{(w_3 * 1) + 1}{(w_3 * 1) + 2} = 0.6$$

$$P(a2 = 2|'false) = \frac{(w_1 * 1) + 1}{(w_1 * 1 + w_2 * 1) + 2} = 0.426$$

$$P(a3 = 3|'true') = \frac{1}{(w_3 * 1) + 3} = 0.286$$

$$P(a3 = 3|'false') = \frac{(w_2 * 1) + 1}{(w_1 * 1 + w_2 * 1) + 3} = 0.415$$

④ According to Eq.(7),

$$P(y_1) = P('true')\prod_{j=1}^{3}P(a_j \mid 'true') + P('false')\prod_{j=1}^{3}P(a_j \mid 'false')$$

$$=0.0496 +0.0526=0.1022$$

⑤ According to Eq.(8)

$$P('true'|y_1) = \frac{P('true')\prod_{j=1}^{3}P(a_j \mid 'true')}{P(y_1)} = 0.485$$

$$P('false'|y_1) = \frac{P('false')\prod_{j=1}^{3}P(a_j \mid 'false')}{P(y_1)} = 0.514$$

$P('true'|y_1) < P('false'|y_1)$, So the test instance y_1's category is false.

So, we can see clearly that calculating weights without considering the number of defects can cause serious mistakes, such as some instances of potentially defective are misjudged as having no defects.

Why did this happen? The reason is that a large number of non-defective instance's weight accumulation will leads to the defective instance's weight has a little impact on the final result. That is to say, the class imbalance caused part of information has little influence on the result, unfortunately it maybe just the part of information we need. This is why we need to consider the addition of the number of defects.

Algorithm1 presents the pseudo-code of NTNB approach.

Algorithm 1. NTNB approach

Input: Training set **L:** $\{x_1, x_2, \ldots, x_m\}$, Test set **U:** $\{u_1, u_2, \ldots, u_n\}$

Output: The classifier P

```
1.  for each attribute a_j in U do:
2.      get the max value and min value for a_j.
3.      get the number of different values n_j for a_j:
4.  end for
5.  for each instance x_i in L do:
6.      calculate w_i through Eq.(4)
7.      get the total number of categories n_c
8.  end for
9.  According to Eq.(5, 6, 7, 8):
10.     Build weighted Naïve Bayes
11. for each instance u_i in U do:
12.     predict u_i through Eq.(1, 2)
13. end for
14. return P
```

IV. EXPERIMENTS

In this section, we evaluate our proposed NTNB approach to perform CCDP. We first introduce the experiment dataset and the performance measures. Then, in order to investigate the performance of NTNB, we perform some contrast experiment.

A. Data set

In this experiment, we employ 6 available and commonly used software project datasets with their 26 releases which can be obtained from PROMISE [7]. The details about the datasets is shown in Table I, where *#Instance* represents the number of instances, *#Defects* represents the total number of faults in the release, *%Defect* represents the percentage of defect-prone instances, and *Max* is the maximum value of faults.

TABLE I. DETAILS OF EXPERIMENT DATASET

Project	*Release*	*#Instance*	*#Defects*	*%Defects*	*Max*
Ant	Ant-1.3	125	33	16.0%	3
	Ant-1.4	178	47	22.5%	3
	Ant-1.5	293	35	10.9%	2
	Ant-1.6	351	184	26.2%	10
	Ant-1.7	745	338	22.3%	10
Camel	Camel-1.0	339	14	3.4%	2
	Camel-1.2	608	522	35.5%	28
	Camel-1.4	872	335	16.6%	17
	Camel-1.6	965	500	19.5%	28
Jedit	Jedit-3.2	272	382	33.1%	45
	Jedit-4.0	306	226	24.5%	23
	Jedit-4.1	312	217	25.3%	17
	Jedit-4.2	267	106	13.1%	10
	Jedit-4.3	492	12	2.2%	2
Synapse	Synapse-1.0	157	21	10.2%	4
	Synapse-1.1	222	99	27.0%	7
	Synapse-1.2	256	145	33.6%	9
Prop	Prop-1	18471	2738	14.8%	37
	Prop-2	23014	2431	10. 6%	22

Project	*Release*	*#Instance*	*#Defects*	*%Defects*	*Max*
	Prop-3	10274	1180	11. 5%	11
	Prop-4	8718	840	9.6%	22
	Prop-5	8516	1299	15.3%	19
	Prop-6	660	66	10%	4
Forrest	Forrest-0.6	6	1	16.7%	1
	Forrest-0.7	29	5	17.2%	8
	Forrest-0.8	32	2	6.3%	4

There are the same 20 independent variables (the 20 feature metrics) and one dependent variable (the number of faults) in each dataset [19].

And we combine the different versions of the same project as a data set. And choose five from them together as a training set, the remaining one as a test set to do experiment.

B. Performance measures

In the experiment, we employ three commonly used performance measures including *pd*, *pf* and *f-measure* to judge the experimental results [5]. They are summarized as follows.

TABLE II. PERFORMANCE MEASURES

		Actual	
		yes	***no***
Predicted	***yes***	***TP***	***FP***
	no	***FN***	***TN***

- TP is the number of defective instances are correctly predicted. FP is the number of defective instances are predicted as non-defective. FN is the number of non-defective instances are predicted as defective. TN is the number of non-defective instances are correctly predicted.

- The precision is the measure of defective modules that are correctly predicted within the instances which is predicted to be defective. The higher the precision, the better the results.

$$\text{precision} = \frac{TP}{TP+FP} \tag{9}$$

- The recall rate or *pd* is the measure of defective modules that are correctly predicted within the defective class. The higher the *pd*, the fewer the false negative results.

$$pd = \frac{TP}{TP+FN} \tag{10}$$

- Probability of false alarm or *pf* is the measure of non-defective modules that are incorrectly predicted within the non-defective class. Unlike *pd*, the lower the *pf* value, the better the results.

$$pf = \frac{FP}{FP+TN} \tag{11}$$

- The *f*-measure is the harmonic average of *pd* and precision. The higher the measure, the better the results.

$$f = \frac{2*pd*\text{precision}}{pd+\text{precision}} \tag{12}$$

Here we generally use *f*-measure as the total performance evaluation standard because it's the combination of *pf* and precision.

C. Experimental results

In order to better verify the performance of NTNB algorithm, we use the training set and test set in table III to do a set of comparative experiments. We mainly compared the *pd*, *pf* and f parameters of NTNB, TNB, and NB. Their respective characteristics are as follows:

NTNB considering the number of defects based on the TNB to avoid the imbalance problem. The imbalance problem is in the process of forecasting, the accumulation of a large number of defective instance's weights resulting in the defective instance's weight has a little impact on the final result.

TNB assigns the data weights according to the similarity between the training instance and the test set, and builds the Naïve Bayes model on these weighted data [5].

While NB did not consider the difference between cross-company data, directly building the model.

The following table III shows the *pf* and *pf* values in the experimental results for our three approaches, where the boldface represents the best result of this experiment. And Fig. 1 shows the scatter plots of *pd* and *pf* for three CCDP models on 6 datasets. If it has a higher recall rate *pd* and less false alarm *pf*, then the defect model has more points distributed in the lower right corner, which also represents a better performance for the forecasting model.

TABLE III. PD,PF VALUES, THE REASULTS OF THREE APPROACH

No.	Test Set	NTNB		TNB		NB	
		PD	PF	PD	PF	PD	PF
1	synapse	**0.556**	0.220	0.506	0.180	0.305	**0.112**
2	prop	0.182	**0.073**	**0.257**	0.118	0.197	0.137
3	jedit	0.696	0.321	**0.706**	0.327	0.423	**0.207**
4	forrest	**0.625**	0.305	0.375	0.237	0.000	**0.017**
5	camel	**0.393**	0.187	0.383	0.184	0.110	**0.099**
6	ant	0.726	0.301	**0.729**	0.300	0.346	**0.137**

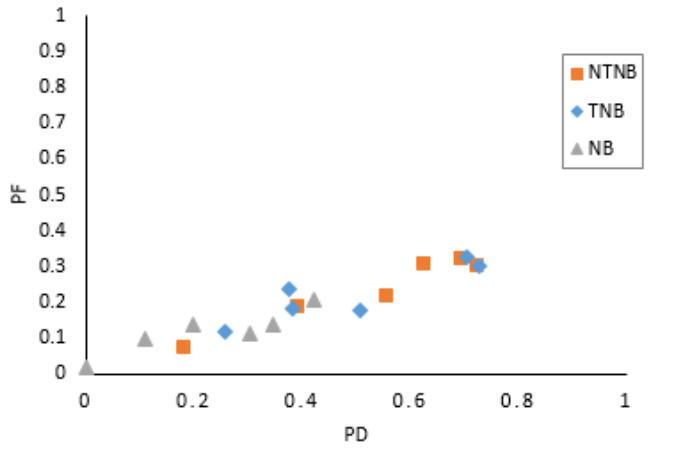


Fig. 1. Scatter plots of *pd* and *pf* for three CCDP models on 6 dataset

The table IV is the *f*-measures in the experimental results for our three approaches, where the boldface represents the best result of this experiment. The experimental results show that in some data sets with unbalanced problems, considering the number of defects information can effectively avoid some defective instance is misjudged as non-defective instance, and improve the accuracy of prediction [20].

TABLE IV. F-MEASURE VALUES, THE REASULTS OF THREE APPROACH

No.	Test data	NTNB	TNB	NB
1	synapse	**0.506**	0.498	0.337
2	prop	0.213	**0.245**	0.151
3	jedit	0.431	**0.432**	0.240
4	forrest	**0.323**	0.240	0.000
5	camel	**0.369**	0.363	0.116
6	ant	0.504	**0.506**	**0.**314

V. CONCLUSION AND FUTURE WORK

In the field of cross-company defects prediction, many models do not consider the defects number information in predicting the result. However, according to our analysis, in some imbalance problems, considering the number of defects will effectively eliminate the impact of a large number of non-defective instance's weight accumulation. To prove this, we conducted a set of comparative experiments on six open cross-company datasets. The results show that considering the number of defects information can effectively avoid some defective instance is misjudged as non-defective instance, and improve the accuracy of prediction in some unbalanced problems. In the following work, we want to find a balance to avoid defects number information overfitting the forecast result. For example, give an extreme example: there are 100000 errors for one instance, then what can we do to avoid it exceeding intervening the result? In addition, we will apply our model on the social data to further prove its effectiveness [21-22], and try to apply our model to other areas [11-12].

ACKNOWLEDGMENT

This work is partly supported by the grants of National Natural Science Foundation of China (No.61572374, No.U163620068, No.U1135005) and the Academic Team Building Plan from Wuhan University and National Science Foundation (NSF) (No. DGE-1522883).

REFERENCES

[1] M. Liu, L. Miao, and D. Zhang, "Two-stage cost-sensitive learning for software defect prediction," IEEE Transactions on Reliability,2014, 63(2):676-686.

[2] X. Y. Jing, S. Ying, Z. W. Zhang, S. S. Wu, and J. Liu, "Dictionary learning based software defect prediction," in: Proc. of the 36th International Conference on Software Engineering (ICSE), 2014,pp. 414-423.

[3] Xiaoyuan Jing et al, "Heterogeneous Cross-Company Defect Prediction by Unified Metric Representation and CCA-Based Transfer Learning," in: Proc. of the 10th Joint Meeting on Foundations of Software Engineering, 2015, pp.496-507.

[4] B. Turhan, T. Menzies, A.B. Bener, J.D. Stefano, "On the relative value of crosscompany and within-company data for defect prediction," Empirical Software Engineering, 2009, 14(5):540-578.

[5] Ying Ma, Guangchun Luo, Xue Zeng and Aiguo Chen, "Transfer learning for cross-company software defect prediction," Information and Software Technology,2012, 54:248–256.

[6] Y. Shi, Z. Lan, W. Liu and W. Bi, "Extending semi-supervised learning methods for inductive transfer learning," In: Ninth IEEE International Conference on Data Mining, 2009, pp.483–492.

[7] G. Boetticher, T. Menzies, T. Ostrand, The PROMISE Repository of Empirical Software Engineering Data, 2007 <http://promisedata.org/repository>.

[8] Xu, Z., et al, " Hierarchy-Cutting Model based Association Semantic for Analyzing Domain Topic on the Web," IEEE Transactions on Industrial Informatics, doi:10.1109/TII.2017.2647986.

[9] Xu, Z., et al, "The Mobile Media based Emergency Management of Web Events Influence in Cyber-Physical Space," Wireless Personal Communications: ,DOI: 10.1007/s11277-016-3689-7.

[10] Xu, Z., et al, "Building Knowledge Base of Urban Emergency Events based on Crowdsourcing of Social Media," Concurrency and Computation: Practice and Experience, 2016 ,28(15) :4038-4052

[11] Ziwei Liu, Chuanbo Wei, Yang Ma, Hui Li, Xiaoguang Niu* and Lina Wang, "UCOR: An Unequally Clustering-based Hierarchical Opportunistic Routing Protocol for WSNs," Springer Berlin Heidelberg, 2013, 7992:175-185.

[12] Ziwei Liu, Xiaoguang Niu, Xu Lin, Ting Huang Yunlong Wu and Hui Li, "A Task-Centric Cooperative Sensing Scheme for Mobile Crowdsourcing Systems," 《Sensors》, 2016, 16(5):746.

[13] Xiao Yu, Jin Liu, Zijiang James Yang, Xiao Liu, Xiaofei Yin and Shijie Y, "Bayesian Network Based Program Dependence Graph for Fault Localization". ISSRE Workshops 2016: 181-188.

[14] Zhou Xu, Jin Liu, Zijiang Yang, Gege An and Xiangyang Jia, "The Impact of Feature Selection on Defect Prediction Performance: An Empirical Comparison". ISSRE 2016: 309-320.

[15] Xiao Yu, Jin Liu, Mandi Fu, Chuanxiang Ma and Guoping Nie, "A Multi-Source TrAdaBoost Approach for Cross-Company Defect Prediction". SEKE 2016: 237-242.

[16] Zhou Xu, Jifeng Xuan, Jin Liu and Xiaohui Cui, "MICHAC: Defect Prediction via Feature Selection Based on Maximal Information Coefficient with Hierarchical Agglomerative Clustering". SANER 2016: 370-381.

[17] L. Peng, B. Yang and Y. Chen, "A. Abraham, Data gravitation based classification," Information Sciences,2009,179(6):809-819.

[18] E. Frank, M. Hall, B. Pfahringer, "Locally Weighted Naive Bayes," in: Proceedings of the Conference on Uncertainty in Artificial Intelligence, 2003, pp. 249–256.

[19] Xiao Yu, Jin Liu, Mandi Fu, Chuanxiang Ma, Guoping Nie: "A Multi-Source TrAdaBoost Approach for Cross-Company Defect Prediction.," SEKE 2016: 237-242

[20] Lin Chen, Bin Fang, Zhaowei Shang and Yuanyan Tang, "Negative samples reduction in cross-company software defects prediction," Information and Software Technology,2015,62:67-77.

[21] Zheng Xu et.a, "Multi-modal Description of Public Safety Events using Surveillance and Social Data," IEEE Transactions on Big Data, 2017, 10.1109/TBDATA.

[22] Zheng Xu, Yunhuai Liu, Hui Zhang, Xiangfeng Luo, Lin Mei and Chuanping Hu, "Building the Multi-Modal Storytelling of Urban Emergency Events Based on Crowd sensing of Social Media Analytics," Mobile Networks and Applications: ,DOI: 10.1007/s11036-016-0789-2.

[23] T.M. Khoshgoftaar, E.B. Allen, J.P. Hudepohl and S.J. Aud, "Application of neural networks to software quality modeling of a very large telecommunications system," IEEE Transactions on Neural Networks,1997, 8 (4): 902–909.

[24] S. Kanmani, V. Rhymend Uthariaraj, V. Sankaranarayanan and P. Thambidurai, "Object-oriented software fault prediction using neural networks," Information and Software Technology,2007, 49 (5) :483–492.

Using Class Imbalance Learning for Cross-Company Defect Prediction

Xiao Yu[1], Mingsong Zhou[2], Xu Chen[1*], Lijun Deng[3], Lu Wang[4]
[1]State Key Lab. of Software Engineering, Computer School, Wuhan University, Wuhan, China
[2]Department of Information Science and Technology, Beijing Normal University, Beijing, China
[3]College of Business, City University of Hong Kong, Kowloon Tong, China
[4]School of Computer Science and Information Engineering, HuBei University, Wuhan, China
*Corresponding author email: xuchen@whu.edu.cn

***Abstract*—Cross-company defect prediction (CCDP) is a practical way that trains a prediction model by exploiting one or multiple projects of a source company and then applies the model to target company. Unfortunately, the performance of such CCDP models is susceptible to the high imbalanced nature between the defect-prone and non-defect classes of CC data. Class imbalance learning is applied to alleviate this issue. Because many class imbalance learning methods have been proposed, there is an imperative need to analyze and compare the performance of these methods for CCDP. Although prior empirical studies have proven AdaBoost.NC algorithm achieves the best performance for defect prediction. This observation leads us to conduct a careful empirical study the issues of if and how class imbalance learning methods can benefit cross-company defect prediction. We investigate different types of class imbalance learning methods, including under-sampling technique, over-sampling technique and over sampling followed by under-sampling technique on the cross-company defect prediction performance over 15 publicly available datasets. Experimental results show that under-sampling technique achieves the best overall performance in terms of the g-measure among those methods we studied.**

Keywords—software defect prediction;cross-company defect prediction; class imbalance learning

I. INTRODUCTION

Software defect prediction is one of the most important software quality assurance techniques. It aims to detect the defect proneness of new software modules via learning from defect data. With the advent of big data era and the development of machine learning techniques [1], many machine learning algorithms are applied to solve the practical problems in life [2-4]. Similarly, many efficient software defect prediction approaches [5-11] using machine learning techniques have been proposed, but they are usually confined to within company defect prediction (WCDP). WCDP works well if sufficient data is available to train a defect prediction model. However, it is difficult for a new company to perform WCDP if there is limited historical data. Cross-company defect prediction (CCDP) is a practical approach to solve the problem. It trains a prediction model by exploiting one or multiple projects of a source company and then applies the model to target company [12].

Due to the increased prevalence of machine learning and transfer learning techniques, a number of CCDP models have been proposed during the past decade [13-21]. Nevertheless, a challenge that threatens the performance of such CCDP models is the high imbalanced nature between the defect-prone and non-defect classes of CC data, i.e., defect datasets with much more non-defect instances (majority) than defect-prone instances (minority). Existing studies [22-23] have shown that the class imbalance problem may cause difficulties for learning, as most classification algorithms only perform optimally when the number of instances of each class is roughly the same. When these algorithms are trained by a highly skewed dataset in which the minority class is heavily outnumbered by the majority class, these classifiers tend to favor the majority class and have less ability to classify the minority class.

Class imbalance learning is applied to alleviate this issue. Because many class imbalance learning methods have been proposed, there is an imperative need to analyze and compare the performance of these methods for CCDP. Although prior empirical studies have proven AdaBoost.NC algorithm achieves the best performance for defect prediction [23]. This observation leads us to conduct a careful empirical study the issues of if and how class imbalance learning methods can benefit cross-company defect prediction.

Therefore, several data sampling works should be done before building the CCDP model. For example, Lin et al. [19] introduced a novel CCDP approach named Double Transfer Boosting (DTB). DTB firstly uses NN filter to filter out irrelevant CC data, and then uses SMOTE algorithm [24] to re-sample the CC data before building the CCDP model.

As the first effort of an in-depth study of class imbalance learning methods in CCDP, this paper explores their potential by focusing on research questions

RQ1: How effective are class imbalance learning methods?

RQ2: Do different class imbalance learning methods have significantly distinct effectiveness on CCDP?

For the two questions, we conduct a large scale empirical study on six class imbalance learning methods and compare them with CCDP models without involving any class imbalance learning method. The six class imbalance learning methods are Random under-sampling (RUS) [25], Near Miss (NM) [26], Synthetic Minority Oversampling Technique (SMOTE) [24]

DOI reference number: 10.18293/SEKE2017-035

and ADASYN [27], SMOTE Tomek links (STOME) [28] and SMOTE ENN (SENN) [29], covering three major types of solutions to learning from imbalanced data, i.e., under sampling technique, over sampling technique and over sampling followed by under-sampling technique. In the experiments, we choose the Naïve Bayes (NB), random forest (RF) and logistic regression (LR) as the CCDP models and pd, pf and g-measure as the evaluation measures. Experimental results over 15 publicly available datasets show that under-sampling technique achieves the best overall performance in terms of the g-measure among those methods we studied.

The remainder of this paper is organized as follows. Section 2 presents the background knowledge about cross-company defect prediction. Section 3 describes the class imbalance learning methods studied in this work. Section 4 presents the data sets, performance measures and the experimental results. Section 5 describes the treats to validity. Finally, Section 6 addresses the conclusion and points out the future work.

II. Related Work

In this section, we briefly review the existing cross-company and cross-company defect prediction approaches.

Briand et al. [12] used logistic regression and MARS models to learn a defect predictor, which is also the earliest work on CCDP. Zimmermann et al. [13] studied CCDP models on 12 real-world applications datasets. Their results indicate that CCDP is still a serious challenge. Turhan et al. [14] investigated the applicability of CC data for building localized defect predictors using 10 projects collected from two different companies including NASA and SOFTLAB. And they have proposed a nearest neighbor (NN) filter to select CC data. He et al. [15] investigates defect predictions in the cross-project context focusing on the selection of training data. Furthermore, they proposed an approach to automatically select suitable training data for projects without historical data so that the results of their experiments are comparable with WCDP, which indicated that some approach of CCDP can comparable to WCDP. They noted that learning predictors using the data from other projects can be a potential way to defect prediction without any historical data. In order to find data for quality prediction, Peters et al. [16] introduced the Peters filter to select training data via the structure of other projects. They compared the filter with two other approaches for quality prediction to assess the performance of the Peters filter, and found that 1) WCDP are weak for small data sets; 2) the Peters filter + CCDP builds better and more useful predictors. Zhang et al. [17] proposed sample-based methods for software defect prediction. For a large software system, they could select and test a small percentage of modules, and then built a defect prediction model to predict defect-proneness of the rest of the modules. They described three methods for selecting a sample and proposed a novel active semi-supervised learning method ACoForest to facilitate the active sampling. The results showed that the proposed methods are effective and have potential to be applied to industrial practice. Ma et al. [18] proposed a novel algorithm called Transfer Naive Bayes (TNB) to transfer cross-company data information into the weights of the training data and then build the predictor based on re-weighted CC data. The results indicated that TNB is more accurate in terms of AUC, within less runtime than the state of the art methods and can effectively achieve the CCDP task. The heterogeneous CCDP (HCCDP) task is that the source and target company data is heterogeneous. Jing et al. [11] provided an effective solution for HCCDP. They proposed a unified metric representation (UMR) for the data of source and target companies and introduced canonical correlation analysis (CCA), an effective transfer learning method, into CCDP to make the data distributions of source and target companies similar. Results showed that their approach significantly outperforms state-of-the-art CCDP methods for HCCDP with partially different metrics and for HCCDP with totally different metrics, their approach is also effective.

Turhan et al. [20] introduced a mixed model of within and cross data for CCDP to investigate the merits of using mixed project data for binary defect prediction. Results showed that when there is limited project history, mixed model for CCDP can achieve good performance which can be comparable to WCDP. It provided a new idea to CCDP that the use of a small amount of labeled WC data would be very valuable to improve the performance of CCDP. Lin et al. [19] introduced a novel approach named Double Transfer Boosting (DTB) to narrow the gap of different distributions between CC data and WC data and to improve the performance of CCDP by reducing negative samples in CC data.

III. Methodology

In this section, we only provide a brief description of the 6 class imbalance learning methods studied in this work due to the space limit. These methods cover three types including under sampling technique, over sampling technique and over sampling followed by under-sampling technique.

A. Under sampling techniques

Under-sampling is a technique to reduce the number of samples in the majority class, where the size of the majority class sample is reduced from the original datasets to balance the class distribution. In this study, we employ two representative under-sampling methods, i.e, Random under-sampling (RUS) and Near Miss (NM).

Random under-sampling (RUS) is a simple method to select a subset of majority class samples randomly and then combine them with minority class sample as a training set. The procedure of random under-sampling is as follows:

1. Calculate the ratio of the minority class to the majority class, and get the sampling frequency.
2. Sample the majority class by the sampling frequency.
3. Select all samples in the minority.
4. Combine selected samples and attributes for training.

Near Miss selects negative examples that are close to some of the positive examples. In the method, we select negative examples whose average distances to three closest positive examples are the smallest. This method guarantee every positive example is surrounded by some negative examples. Finally, in selection of most distant negative examples, we

choose the negative examples whose average distances to the three closest positive examples are the farthest. We expect the Near Miss methods should perform better than the random and distant methods, and the random method should work better than the distant method. We also expect that Near Miss method should achieve high precision and low recall while the distant method should achieve high recall and low precision.

B. Over sampling techniques

Over-sampling is a technique in which the minority class is over-sampled by creating "synthetic" examples rather than by under-sampling with replacement. In this study, we employ two representative under-sampling methods, i.e, Synthetic Minority Oversampling Technique (SMOTE) and ADASYN.

The procedure of SMOTE is as follows:

1. For each instance in the minority class, calculate the Euclidean distance between it and other samples in the minority class to find its k nearest neighbor.

2. According to the amount of over-sampling, determine the sampling rate and select a certain number of samples from k nearest neighbor randomly.

3. Take the difference of the feature vector between it and its nearest neighbor.

4. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration.

5. Generate new samples for each instance in the minority class and add new samples into it.

ADASYN is based on the idea of adaptively generating minority data samples according to their distributions: more synthetic data is generated for minority class samples that are harder to learn compared to those minority samples that are easier to learn. The procedure of ADASYN is as follows:

1. Calculate the degree of class imbalance.
2. If degree is less than *d* then (*d* is a preset threshold for the maximum tolerated degree of class imbalance ratio).
3. Calculate the number of synthetic data examples that need to be generated for the minority class.
4. For each example, find *K* nearest neighbors based on the Euclidean distance in *n* dimensional space.
5. Normalize according to a density distribution.
6. Calculate the number of synthetic data examples that need to be generated for each minority example.
7. For each minority class data example, generate synthetic data examples according to the following steps:

   Do the loop:

   i. Randomly choose one minority data example from the *k* nearest neighbors for data.

   ii. Generate the synthetic data example.

C. Over sampling followed by under sampling

Over sampling followed by under-sampling is a technique in which the minority class is over-sampled by creating "synthetic" examples followed by under-sampling with replacement. In this study, we employ two representative under-sampling methods, i.e, SMOTE Tomek links (STOME) and SMOTE ENN (SENN).

Although over-sampling minority class examples can balance class distributions, some other problems usually present in data sets with skewed class distributions are not solved. Frequently, class clusters are not well defined since some majority class examples might be invading the minority class space. The opposite can also be true, since interpolating minority class examples can expand the minority class clusters, introducing artificial minority class examples too deeply in the majority class space. Inducing a classifier under such a situation can lead to overfitting. In order to create better-defined class clusters, we propose to apply Tomek links to the over-sampled training set as a data cleaning method. Thus, instead of removing only the majority class examples that form Tomek links, examples from both classes are removed. The application of this method can be illustrated as follows. First, the original data set is over-sampled with Smote, and then Tomek links are identified and removed, producing a balanced data set with well-defined class clusters. The STOME links method was first used to improve the classification of examples for the problem of annotation of proteins in Bioinformatics.

The motivation behind SENN is similar to STOME. SENN tends to remove more examples than the Tomek links does, so it is expected that it will provide a more in depth data cleaning. Differently from NCL which is an under-sampling method, ENN is used to remove examples from both classes. Thus, any example that is misclassified by its three nearest neighbors is removed from the training set.

IV. Experiments

In this section, we first introduce the experiment dataset and the performance measures. Then, in order to investigate the performance of class imbalance learning methods, we perform some empirical experiments to find answers to the research questions mentioned above.

A. Data set

In this experiment, we employ 15 available and commonly used datasets which can be obtained from PROMISE. The 15 datasets have the same 20 attributes, so we can apply all attribute information directly. Table 1 tabulates the details about the datasets.

TABLE I. Details of Experiment Dataset

Project	*Examples*	*%Defective*	*Description*
ant	125	16	Open-source
arc	234	11.5	Academic
camel	339	3.8	Open-source
elearn	64	7.8	Academic
jedit	272	33.1	Open-source
log4j	135	25.2	Open-source
lucene	195	46.7	Open-source
poi	237	59.5	Open-source

Project	*Examples*	*%Defective*	*Description*
prop	660	10	Proprietary
redaktor	176	15.3	Academic
synapse	157	10.2	Open-source
systemdata	65	13.8	Open-source
tomcat	858	9	Open-source
xalan	723	15.2	Open-source
xerces	162	47.5	Open-source

B. Performance measures

In the experiment, we employ three commonly used performance measures including *pd*, *pf* and *g-measure*. They are defined in Table 2 and summarized as follows.

TABLE II. PERFORMANCE MEASURES

		Actual	
		yes	***no***
Predicted	***yes***	***TP***	***FP***
	no	***FN***	***TN***
pd	$\frac{TP}{TP+FN}$		
pf	$\frac{FP}{FP+TN}$		
g-measure	$\frac{2*pd*(1-pf)}{pd+(1-pf)}$		

- Probability of detection or *pd* is the measure of defective modules that are correctly predicted within the defective class. The higher the *pd*, the fewer the false negative results.

- Probability of false alarm or *pf* is the measure of non-defective modules that are incorrectly predicted within the non-defective class. Unlike *pd*, the lower the *pf* value, the better the results.

- *g-measure* is a trade-off measure that balances the performance between *pd* and *pf*. A good prediction model should have high *pd* and low *pf*, and thus leading to a high *g-measure*.

C. Experimental Procedure

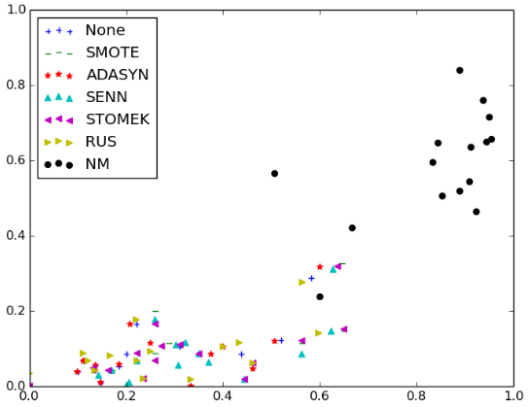
In every experiment, one dataset is selected as WC data and the rest are regarded as CC data to conduct the experiment. The CC data is considered as basic training data which will be adjusted in every experiment. All processing steps (data filter and data sampling) are done on CC data. Then processed CC data are used to build the CCDP model. Finally, the resulting model is evaluated on the WC data. The procedure will be repeated 30 times in every experiment to avoid sample bias. Then, the mean values of performance are calculated.

In this experiment, we choose three representative classifiers as the basic prediction model, Naive Bayes (NB), Random Forest (RF), and Logistic Regression (LR).The reason we choose these classifiers is that these classifiers fall into three different families of learning methods. NB is a probabilistic classifier; RF is a decision-tree classifier; and LR is a linear model for classification.

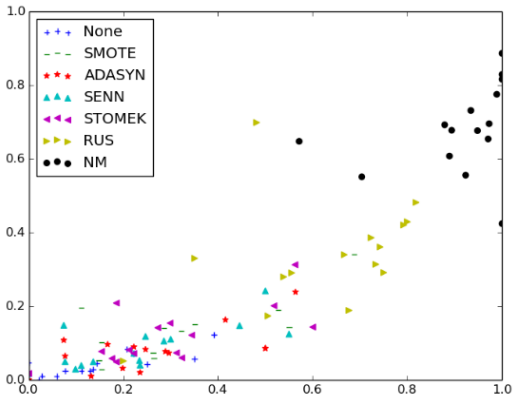
D. Experimental results

Fig. 1 presents the scatter plots of (PD, PF) points from the seven training methods on the ten SDP data sets. We can gain the following results from Fig. 1.

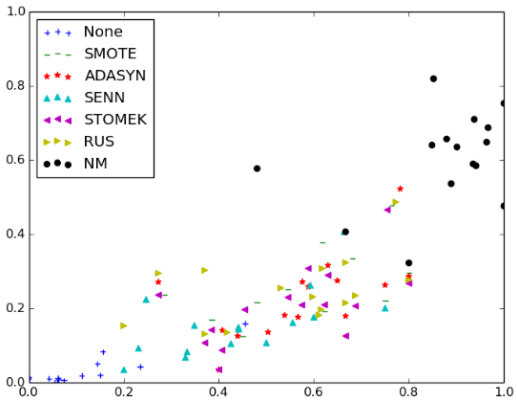
Although CCDP models without involving any class imbalance learning algorithms appears to be better at PF than other class imbalance learning models, it has lowest PD value, which proves class imbalance learning should be applied to CCDP models. In terms of the defect detection rate (PD), NM outperforms other class imbalance learning models, which shows its effectiveness in finding defects. However, NM has relatively high PF value. In terms of the false alarm rate (PF), although SENN is the best, it performs not well in PD, which makes it hardly useful in practice.



(a)NB



(b)RF



(c)LR

Fig. 1. Performances with Scatter plots of (PD, PF) points of the six class imbalacne learning methods and the CCDP model without class imbalance learning on the fifteen projects

Table 3-5 also show the g-measure values for each project on all the methods. From the tables, we can find that CCDP models without involving any class imbalance learning algorithms perform the worst in g-measure, which proves that the class imbalance problem is a big challenge for CCDP. In terms of NB model, NM achieves the best average g-measure value among the 15 projects. In terms of RF, RUS achieves the best average g-measure value among the 15 projects. In terms of LG, ADASYN achieves the best average g-measure value among the 15 projects. RUS and NM are under-sampling techniques, which reduce the number of CC instances in the non-defect class. We guess the reason under-sampling technique performs better is that it retains the defect-prone instances.

TABLE III. G-MEASURE PERFORMANCES WITH NAVIE BAYES

Project	*None*	*SMOTE*	*ADASYN*	*SENN*	*STOME*	*RUS*	*NM*
ant	0.32	0.50	0.38	0.50	0.50	**0.55**	0.43
arc	0.30	0.35	0.30	**0.53**	0.40	0.35	0.62
camel	0.62	0.61	0.62	0.46	0.61	0.61	**0.67**
elearn	0.00	0.00	0.00	0.33	0.00	0.00	**0.67**
jedit	0.46	0.47	0.55	0.47	0.46	**0.58**	0.51
log4j	0.25	0.37	0.25	0.34	0.37	0.37	**0.62**
lucene	0.17	0.23	0.17	0.24	0.23	0.21	**0.52**
poi	0.23	0.28	0.23	0.28	0.27	0.23	**0.49**
prop	0.23	0.43	0.23	0.45	0.41	0.28	**0.54**
redaktor	0.19	**0.40**	0.19	0.35	0.35	0.19	0.27
synapse	0.59	0.68	0.53	**0.69**	0.68	0.39	0.38
system	0.50	0.61	0.50	0.61	0.61	0.49	**0.62**
tomcat	0.65	**0.73**	0.64	0.72	0.73	0.70	0.60
xalan	0.64	**0.65**	0.63	**0.65**	0.65	0.63	0.50
xerces	0.34	0.39	0.33	0.39	0.39	0.34	**0.46**
AVG	0.37	0.45	0.37	0.47	0.44	0.39	**0.53**

TABLE IV. G-MEASURE PERFORMANCES WITH RANDOM FOREST

Project	*None*	*SMOTE*	*ADASYN*	*SENN*	*STOME*	*RUS*	*NM*
ant	0.51	0.67	0.64	**0.67**	0.70	0.66	0.29
arc	0.19	0.25	0.44	0.19	0.31	**0.62**	0.54
camel	0.14	0.26	0.14	0.14	0.26	**0.61**	0.60
elearn	0.00	0.00	0.00	0.00	0.00	0.33	**0.73**
jedit	0.25	0.47	0.44	0.44	0.49	**0.70**	0.36
log4j	0.11	0.41	0.37	0.37	0.48	**0.73**	0.51
lucene	0.04	0.26	0.23	0.17	0.29	**0.62**	0.41
poi	0.05	0.40	0.32	0.37	0.46	**0.66**	0.47
prop	0.23	0.34	0.28	0.23	0.34	**0.68**	0.45
redaktor	0.00	0.19	0.13	0.13	0.30	0.37	0.20
synapse	0.39	0.67	0.64	0.60	0.61	**0.72**	0.31
system	0.35	0.35	0.35	0.35	0.35	**0.66**	0.54
tomcat	0.33	0.49	0.55	0.43	0.44	**0.66**	0.48
xalan	0.54	0.63	**0.64**	0.58	0.62	0.63	0.46
xerces	0.22	0.42	0.38	0.38	0.41	**0.46**	0.43
AVG	0.20	0.39	0.37	0.34	0.40	**0.56**	0.45

TABLE V. G-MEASURE PERFORMANCES WITH LOGISTIC REGRESSION

Project	*None*	*SMOTE*	*ADASYN*	*SENN*	*STOME*	*RUS*	*NM*
ant	0.26	0.74	0.75	**0.77**	0.76	0.76	0.39
arc	0.13	0.52	0.55	0.48	0.52	0.51	**0.62**
camel	0.00	0.52	0.64	0.36	0.53	**0.69**	0.68
elearn	0.00	0.56	0.56	0.33	0.56	0.32	**0.73**
jedit	0.25	0.70	0.67	**0.69**	**0.69**	**0.69**	0.47
log4j	0.11	0.62	**0.65**	0.58	0.63	**0.65**	0.57
lucene	0.10	**0.59**	0.58	0.48	0.56	0.56	0.51
poi	0.08	**0.67**	0.63	0.57	0.66	**0.67**	0.49
prop	0.11	0.63	**0.64**	0.49	0.63	0.61	0.50
redaktor	0.00	**0.66**	0.65	0.65	**0.66**	0.48	0.29
synapse	0.11	**0.76**	0.74	0.64	0.73	0.72	0.44
system	0.19	**0.75**	0.73	0.66	**0.75**	0.72	0.61
tomcat	0.37	0.59	**0.68**	0.58	0.58	0.67	0.57
xalan	0.59	**0.62**	0.59	**0.62**	**0.62**	0.61	0.51
xerces	0.26	0.41	0.39	0.37	0.40	0.39	**0.45**
AVG	0.17	0.62	**0.63**	0.55	0.62	0.60	0.52

RQ1 Summary. According to the experiment results in Table 3-5, we conclude that these class imbalance learning algorithms can yield better prediction results than CCDP models without involving any class imbalance learning algorithms.

RQ2 Summary. According to the experiment results in Table 3-5, the above observations show that the effectiveness of these class imbalance learning methods exhibits significant differences on the performance of CCDP. Among the three types of imbalance learning methods, under-sampling technique is the winner according to g-measure.

V. THREATS TO VALIDITY

In this section, we discuss several validity threats that may have an impact on the results of our studies.

A. External validity

Threats to external validity occur when the results of our experiments cannot be generalized. As a preliminary result, we performed our experiments on the 15 datasets to answer the research questions. Although these datasets have been widely used in many software defect prediction studies, we still cannot claim that our conclusions can be generalized to other software projects. Nevertheless, this work provides a detail experimental description, including parameter settings (default parameter settings specified by sklearn), thus other researchers can easily replicate our method on new datasets.

B. Internal validity

In our study, we repeat 30 times to avoid sample bias, and calculate average results to verify the performance of all test methods. In this work, we only use three classifiers, Naive Bayes (NB), Random Forest (RF), and Logistic Regression (LR) due to its popularity in defect prediction.

C. Construct validity

In our experiments, we mainly use pd, pf, g-measure to measure the effectiveness of the proposed method. Nevertheless, other evaluation measures such as AUC measure can also be considered.

VI. Conclusion and future work

The cross-company defect prediction is an interest problem in the field of software engineer. The class imbalance problem of defect datasets usually makes it difficult to build a CCDP model with high performance. In this paper, we address the issue how class imbalance learning methods can contribute to CCDP. We conduct a larger scale empirical study to investigate the impact of 6 class imbalance learning methods on the CCDP performance. We conduct experiments on the 15 datasets to evaluate the performance of the methods. The experimental results indicate that under-sampling technique is the winner according to g-measure among the three types of imbalance learning methods.

In the future, we would like to validate the generalization of our conclusion on more company data. In addition, we plan to apply our method to more real-life systems [30-31] to predict the defective module.

Acknowledgment

This work is partly supported by the grants of National Natural Science Foundation of China (No.61572374, No.U163620068, No.U1135005) and the Academic Team Building Plan from Wuhan University and National Science Foundation (NSF) (No. DGE-1522883).

References

[1] Xu Z, Liu Y, Mei L, et al. Semantic based representing and organizing surveillance big data using video structural description technology[J]. Journal of Systems and Software, 2015, 102: 217-225.

[2] Xu Z, Zhang S, Choo K K, et al. Hierarchy-cutting model based association semantic for analyzing domain topic on the web[J]. IEEE Transactions on Industrial Informatics, 2017.

[3] Xu Z, Mei L, Lu Z, et al. Multi-modal Description of Public Security Events using Surveillance and Social Data[J]. IEEE Transactions on Big Data, 2017.

[4] Xu Z, Liu Y, Mei L, et al. The mobile media based emergency management of web events influence in cyber-physical space[J]. Wireless Personal Communications, 2016: 1-14.

[5] Elish K O, Elish M O. Predicting defect-prone software modules using support vector machines[J]. Journal of Systems and Software, 2008, 81(5): 649-660.

[6] Zheng J. Cost-sensitive boosting neural networks for software defect prediction[J]. Expert Systems with Applications, 2010, 37(6): 4537-4543.

[7] Sun Z, Song Q, Zhu X. Using coding-based ensemble learning to improve software defect prediction[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012, 42(6): 1806-1817.

[8] Wang S, Yao X. Using class imbalance learning for software defect prediction[J]. IEEE Transactions on Reliability, 2013, 62(2): 434-443.

[9] Liu M, Miao L, Zhang D. Two-stage cost-sensitive learning for software defect prediction[J]. IEEE Transactions on Reliability, 2014, 63(2): 676-686.

[10] Jing X Y, Ying S, Zhang Z W, et al. Dictionary learning based software defect prediction[C]//Proceedings of the 36th International Conference on Software Engineering. ACM, 2014: 414-423.

[11] Jing X, Wu F, Dong X, et al. Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning[C]//Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. ACM, 2015: 496-507.

[12] Briand L C, Melo W L, Wust J. Assessing the applicability of fault-proneness models across object-oriented software projects[J]. IEEE transactions on Software Engineering, 2002, 28(7): 706-720.

[13] Zimmermann T, Nagappan N, Gall H, et al. Cross-project defect prediction: a large scale experiment on data vs. domain vs. process[C]//Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering. ACM, 2009: 91-100.

[14] Turhan B, Menzies T, Bener A B, et al. On the relative value of cross-company and within-company data for defect prediction[J]. Empirical Software Engineering, 2009, 14(5): 540-578.

[15] He Z, Shu F, Yang Y, et al. An investigation on the feasibility of cross-project defect prediction[J]. Automated Software Engineering, 2012, 19(2): 167-199.

[16] Peters F, Menzies T, Marcus A. Better cross company defect prediction[C]//Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on. IEEE, 2013: 409-418.

[17] Zhang F, Mockus A, Keivanloo I, et al. Towards building a universal defect prediction model[C]//Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014: 182-191.

[18] Ma Y, Luo G, Zeng X, et al. Transfer learning for cross-company software defect prediction[J]. Information and Software Technology, 2012, 54(3): 248-256.

[19] Chen L, Fang B, Shang Z, et al. Negative samples reduction in cross-company software defects prediction[J]. Information and Software Technology, 2015, 62: 67-77.

[20] Turhan B, Mısırlı A T, Bener A. Empirical evaluation of the effects of mixed project data on learning defect predictors[J]. Information and Software Technology, 2013, 55(6): 1101-1118.

[21] Xiao Yu,Jin Liu, Mandi Fu, et al. A Multi-Source TrAdaBoost Approach for Cross-Company Defect Prediction[C]// The 28th International Conference on Software Engineering & Knowledge Engineering. San Francisco Bay, California, USA,2016: 237-242.

[22] Ozturk M M, Zengin A. HSDD: a hybrid sampling strategy for class imbalance in defect prediction data sets[C]//Future Generation Communication Technologies (FGCT), 2016 Fifth International Conference on. IEEE, 2016: 60-69.

[23] Wang S, Yao X. Using class imbalance learning for software defect prediction[J]. IEEE Transactions on Reliability, 2013, 62(2): 434-443.

[24] Chawla N V, Bowyer K W, Hall L O, et al. SMOTE: synthetic minority over-sampling technique[J]. Journal of artificial intelligence research, 2002, 16: 321-357.

[25] Tahir M A, Kittler J, Yan F. Inverse random under sampling for class imbalance problem and its application to multi-label classification[J]. Pattern Recognition, 2012, 45(10): 3738-3750.

[26] Mani I, Zhang I. kNN approach to unbalanced data distributions: a case study involving information extraction[C]//Proceedings of workshop on learning from imbalanced datasets. 2003.

[27] He H, Bai Y, Garcia E A, et al. ADASYN: Adaptive synthetic sampling approach for imbalanced learning[C]//Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on. IEEE, 2008: 1322-1328.

[28] Batista G E, Bazzan A L C, Monard M C. Balancing Training Data for Automated Annotation of Keywords: a Case Study[C]//WOB. 2003: 10-18.

[29] Batista G E, Prati R C, Monard M C. A study of the behavior of several methods for balancing machine learning training data[J]. ACM Sigkdd Explorations Newsletter, 2004, 6(1): 20-29.

[30] Liu Z, Wei C, Ma Y, et al. UCOR: an unequally clustering-based hierarchical opportunistic routing protocol for WSNs[C]//International Conference on Wireless Algorithms, Systems, and Applications. Springer Berlin Heidelberg, 2013: 175-185.

[31] Liu Z, Niu X, Lin X, et al. A Task-Centric Cooperative Sensing Scheme for Mobile Crowdsourcing Systems[J]. Sensors, 2016, 16(5): 746.

Improving Bug Triage with Relevant Search

Xinyu Peng, Pingyi Zhou, Jin Liu*, Xu Chen
State Key Lab of Software Engineering
Computer School, Wuhan University
Wuhan, China
*Corresponding author
{ pengxinyu, zhou_pinyi, jinliu, xuchen}@whu.edu.cn

***Abstract*—Bug triage is a process where bugs are assigned to developers. In large open source projects such as Mozilla and Eclipse, bug triage is time-consuming because numerous bugs are submitted everyday. To improve bug triage, many studies have proposed automatic approaches to recommend proper developers for resolving bugs. These approaches are based on machine learning algorithms, which treat bug triage like text classification. Although they are effective, the accuracy of them can be further improved. Our goal is to propose a method not only has good performance but also is simple. We propose a method based on relevant search technique to recommend developers for the given bugs. First, we construct an index for bugs to make them searchable. Then, for a given bug to be assigned, we utilize the index to search for the bugs related to it. Finally, we analyze these related bugs and recommend developers based on them. We conduct experiments on bugs of Mozilla and Eclipse to evaluate our method. The results indicate that our method has a good performance and outperforms machine learning algorithms like Naïve Bayes and SVM.**

Keywords—bug triage; bug report assignment; issue tracking; problem tracking

I. INTRODUCTION

In software maintenance, bug resolution plays an important role. To effectively manage bugs and coordinate efforts, large open source projects often incorporate bug tracking systems. Bugzilla is a typical example of bug tracking systems, which proposed by Mozilla and adopted by many famous open source projects like Eclipse. Bugs in bug tracking systems are represented by bug reports, which are documents with many fields to record information about the corresponding bugs. The developers in projects depend on the bug reports to manage bugs and exchange opinions.

Figure 1 shows an Eclipse bug report. It has freeform text such as summary and description, which demonstrate what the bug is about and the steps to reproduce the bug. It has some category fields such as product and component. This bug belongs to product JDT and component UI. If a bug has been assigned to a developer like this one, the field named assigned to would refer to the developer. In this case, the developer is Claude Knaus. Besides, status field indicates the stage of the bug in the life-cycle of bug reports. When a bug report is submitted, its status is UNCONFIRMED. The status changes to NEW after a developer verify it. The same developer is often responsible for finding an appropriate developer to fix the bug. After that, the status changes to ASSIGNED. If the bug has been fixed and been verified, its status changes to RESOLVED and VERIFIED or CLOSED respectively. The resolution of a bug in status field could be any among FIXED, DUPLICATE, WORKFORME, WONTFIX, INVALID. For example, the status in Figure 1 is VERIFIED FIXED.

Product: JDT **Component**: UI

Version: 2.0 **Platform**: PC Windows 2000

Severity: normal **Priority**: P3

Status: VERIFIED FIXED

Assigned to: Claude Knaus

Summary: Template - pressing new presents an error

Description: go to Preferences->Java->Templates - press the new button - you get an error saying that the template name must not be null. This is annoying since I didn't have the change to specify one.

Figure 1. Bug Report #4353 of Eclipse

The process where a developer determine which developer is most appropriate to a bug and assigned it to the developer is called bug triage. If a bug is assigned to an improper developer, it leads to the bug re-assignment. The probability of a bug been fixed decreases as the number of re-assignment increases [3]. As the open source projects become more complicated, the number of bug reports submitted everyday constantly increase. It is inefficient that all bug reports are manually assigned. The software development is delayed by the inefficacy. Besides, it gives users an impression that developers are unresponsive and disregard the users' bug reports. This terrible impression will destroy the project's community [1]. To improve the efficiency of bug triage, several studies have proposed automatic methods to recommend appropriate developers for any given bug [1] [2]. These methods are based on machine learning algorithms. They treat bug triage as text classification. Although the methods are effective, the accuracy of them can be further improved.

In this paper, we propose a method utilizing relevant search technique. Our method essentially builds a search application for bug reports. With this search application, we can find relevant bug reports for a given bug then analyze them to get the corresponding developers. To make the bug reports searchable, we first construct an index for them. Given a new bug, we build

DOI reference number: 10.18293/SEKE2017-041

a query based on its summary and description and narrow the search scope with its product and component. Based on the search results, we use their relevance scores to rank developers corresponded to the search results of bug reports. The developers in top rank are recommended for the given bug. Lucene is used to implement the search application.

This paper makes two contributions:

1.It proposes a new relevant search based method to recommend developers for bug triage.

2.It conducts experiments to evaluate the approach on two datasets: Eclipse and Mozilla.

The remainder of the paper is organized as follows. Section 2 demonstrates our approach. Section 3 presents the experiment results. Section 4 discusses our method. Section 5 describes related work. Section 6 concludes this paper.

II. APPROACH

Figure 2 shows our overall framework of bug triage. The framework consists of five steps. It first extracts necessary information such as summary, description, product and component from the original bug reports in step 1. Then it indexes the extracted information in step 2. When we need to recommend developers for a new bug report, it first executes the same information extraction and indexing steps like step 1 and step 2 but just for the given report in step 3 and step 4. Then it searches for the similar bug reports in step 5. Last it uses the searching results to rank the developers and generates a list of them as the recommendation in step 6.

A. Data Preprocessing

The original bug reports have much information that our method does not need. We extracted the necessary information for our method: summary, description, product, component, fixer. Because summary and description are unstructured text, to eliminate data noise, we remove stop words and pure numbers, which do not contain useful information representing the bug reports, then stem them by using Porter stemmer. We join the summary and the description together as content after all.

B. Indexing

To make the searching process possible, we first indexing bug reports. In the process of indexing bug reports, they are treated as documents consisting of fields. Specifically, a bug report is a document having fields like content, product, component. Moreover, the original texts of bug reports are tokenized to terms and transformed to a data structure called inverted index. An inverted index is a sorted list of terms. Each term is associated with pointers linked to documents containing the term. Because the inverted index is sorted and have the pointers for each term, it is efficient for locating a term in the inverted index and finding which documents have the term. It likes the index of books. If we are interested in a subject of a book, we can directly look for the terms about the subject in the index, and go to the related pages without scan through the entire book.

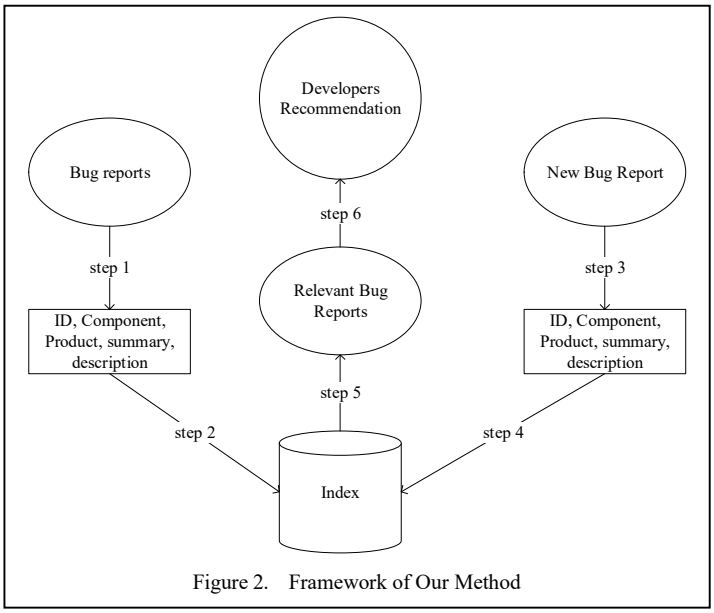


Figure 2. Framework of Our Method

C. Searching

Given a bug to be assigned, we first index it, then construct a query based on its content. Besides, we use its component and product to narrow the searching scope for reducing the noise. At first, we only consider bugs have same component and product as the current bug. If the bug is the first one in its component and product, there are no search results. In this situation, we consider bugs only have the same product. If the bug even is the first one in its product, there are still no search results. Finally, to ensure results exist, we search for related bugs no matter what their component and product are.

The searching results are returned with scores, which represent the relevance between the current bug and others. For consistent with the relevant search context, we use query q and document d_i denote the current bug and one bug of the results respectively. The formula below is used to compute these scores.

$$score(q,d_i) = corrd(q,d_i)\times queryNorm(q)\times \sum_{t\ in\ q}\left(tf(t\ in\ d_i)\times idf(t)^2\times t.getBoost\times norm(t,d_i)\right) \quad (1)$$

In the Equation 1, $tf(t\ in\ d_i)$ stands for the term's frequency, defined as the number of times term t appears in the current document d_i . $idf(t)$ stands for inverse document frequency. This value correlates to the inverse of $docFreq$(the number of documents in which the term t appears).

$coord(q,d_i)$ is a score factor based on how many of the query terms are found in the specified document.

$queryNorm(q)$ is computed as Equation 2, which is a normalizing factor used to make scores between queries comparable. However, this factor does not affect document ranking, since all ranked documents returned by one query are multiplied by the same factor.

$$queryNorm(q) = \frac{1}{\sqrt{\left(q.getBoost^2\times\sum_{t\ in\ q}(idf(t)\times t.getBoost)^2\right)}} \quad (2)$$

In the Equation 1 and 2, $t.getBoost$ is the weight of term t which is specified in the query. It can be changed to make some term more important than others. However, we treat all terms equal and did not change the weight. $q.getBoost$ is the weight of query q, which is used to control the importance of queries and we did not change it too.

$$norm(t, d_i) = lengthNorm \times \prod_{field\ f\ in\ d\ named\ as\ t} f.boost \quad (3)$$

$norm(t, d_i)$ is computed as Equation 3. $f.boost$ is the weight of the field f, which play the same role as the two boost before. We did not change it neither. $lengthNorm$ is a factor in accordance with the number of tokens in a field, shorter fields contribute more to the score.

D. Ranking

After searching, a bunch of related bugs ranked by the scores is returned. Assuming the $score(q, d_i)$ and $score_{max}$ represent the score of ith bug and the max score of all results respectively, we normalize the scores according to the Equation 4, which results in normalized scores within the range of [0, 1].

$$score_{norm}(q, d_i) = score(q, d_i) \div score_{max} \quad (4)$$

Because each bug in the results corresponds to a developer who fixed it. After analyzing searching results, we can get a set of developers. Assuming D denotes the set, dev_i denotes the score of the ith developer in D and A_i denotes the set of bugs in searching results which are assigned to the ith developer, we computed the dev_i as in the Equation 5.

$$dev_i = \sum_{d_i\ in\ A_i} score_{norm}(q, d_i) \quad (5)$$

After computing the scores of the developers in D, we rank them by their scores. The top developers in the rank are recommended for the bug to be assigned.

III. Experiments And Results

A. Dataset

We evaluated our methods on two datasets from two open source software: Eclipse and Mozilla. All data were collected from the websites of their bug tracking systems. For Eclipse, we collected bug reports from 2001-10-11 to 2007-12-14. We drop bug reports before the bug whose id is 4354 because their descriptions are discussions among developers, which are not actual descriptions added when the bug reports are created. Besides, these bug reports were submitted in a very short time, which indicates they were migrated from other bug tracking system. For Mozilla, we collected bug reports from 1998-04-07 to 2008-08-11. We retained bug reports with status CLOSED and FIXED as prior studies did [2], [3], [6], [9].

We extracted bug fixers by checking the "assigned to" fields in the bug reports following previous studies. However, the "assigned to" fields in many bug reports are set to generic names which do not correspond to real people [19]. In Eclipse, bug reports are assigned to generic names like "JDT-UI-Inbox", "JDT-Text-Inbox", "JDT-Core-Inbox". In Mozilla, bug reports are assigned to "nobody". Because these generic names do not represent real developers, we do not recommend them and exclude bug reports whose fixers are among them from the data. Then to reduce noise, bug reports whose fixers appear less than ten times are excluded [19].

After above steps, 91251 bug reports, 650 fixers, 72 products and 450 components are left in Eclipse dataset, while 100964 bug reports, 777 fixers, 59 products and 492 components are left in Mozilla dataset.

To make our experiments are more like the situation in practice, we adopt the longitudinal data setup in [9] [19]. We perform a 10-round incremental analysis on the two project datasets. The bug reports extracted from the two projects are first sorted in chronological order of creation time and then divided into 11 equally-size folds. We form 10 rounds evaluation with the 11 folds. First, in round 0, we create an index using bug reports in fold 0, and we update the index and evaluate our method using the first bug, and then update the index and evaluate our method using the second bug report, and so on for all bug reports in fold 1. Then, in round 1, we proceed in a similar way like fold 1 to test using bug reports in fold 2, and so on. After round 10, we compute the average evaluation metric among all rounds.

To make our results comparable, we choose a metric called Recall@K used in many prior studies [5] [19]. Recall@K is the proportion of bugs whose associated developers is ranked in the top k (k = 1, 3, 5) of the returned results. Given a bug report, if the top k results contain the developer who fixed the bug, we consider the developer is located. So Recall@K of all test bugs equal to the proportion of how many recommendations contain the actual fixer.

B. Research Questions

In this paper, we are interested in the following research questions:

RQ1: How is our method effective compared with other baselines?

To answer the question, we compare our method with the existing machine learning based developer recommendation methods, such as those based on Naïve Bayes and SVM [1], [2], [12]. We use the scikit-learn package to implementation Naïve Bayes and SVM respectively.

RQ2: How does the performance of our method change with the recommendation list increase?

To answer this question, we evaluated our method varying different recommendation list size from 1 to 10. We can see the trends after drawing the evaluated results.

Meanwhile, we used coverage rate to measure the best score our method can reach. The coverage rate is computed like Recall@K, but it considers all developers instead of Top K developers. So it gives the upper bound for our method. Because only if there exist the actual fixers in the search results, our method has the chance to pick them out, which means Recall@K is always smaller than or equal to the coverage rate.

RQ3: How does product and component information influence the effectiveness of our method?

TABLE I. EVALUATION RESULTS OF TWO PROJECTS

Project	Rank	SVM	Naïve Bayes	Our Method
Eclipse	Top1	0.307	0.186	0.438
	Top3	0.518	0.258	0.725
	Top5	0.613	0.286	0.841
Mozilla	Top1	0.223	0.147	0.333
	Top3	0.426	0.227	0.551
	Top5	0.518	0.293	0.634

To figure out the influence of the product and component combinations over our method, we compare the results between our method with and without narrowing the search scope using the product and component information.

C. Results

RQ1: How is our method effective compared with other baselines?

Table I compares the performance of our method with baselines in terms of Recall@1, Recall@3, and Recall@5. For Eclipse, our method achieves average Recall@1 value 0.438, which means that for 43.8% bugs, it successfully recommends their associated developers as top 1. The Recall@5 value is 0.841, which means that for 73.85% bugs, their developers can be found in the top 5 return results. For Mozilla, our method achieves Recall@1 and Recall@5 values 0.333 and 0.634, respectively. It obtains better results than the conventional machine learning based recommendation methods. Comparing with Naïve Bayes, the results of our method are 116% ~ 194% better. Comparing with SVM, the results of our method are 22% ~ 49% better.

TABLE II. COVERAGE RATE OF TWO PROJECTS

Project	metric	without feature	with feature
Eclipse	Coverage rate	0.907	0.953
Mozilla	Coverage rate	0.821	0.795

RQ2: How does the performance of our method change with the recommendation list increase?

Figures 3, 4 present the values from Recall@1 to Recall@10 of our method with coverage rate for Eclipse and Mozilla respectively. We notice that Recall@K values increase along with K values increasing and tend to be stable finally. The Recall@K values are close to the coverage rate from Recall@8, which means the actual fixers are in the Top 8 list already and increasing size of recommendation list is almost useless.

RQ3: How does product and component information influence the effectiveness of our method?

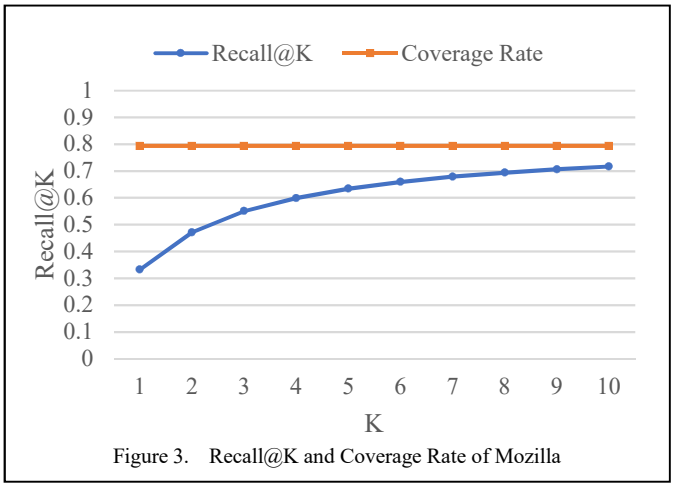


Figure 3. Recall@K and Coverage Rate of Mozilla

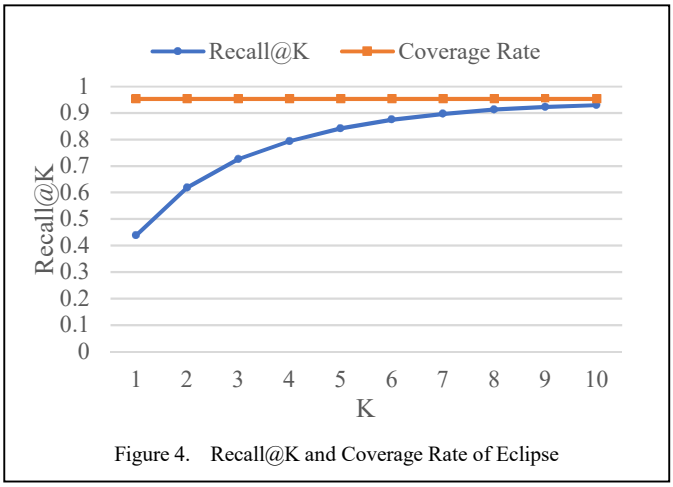


Figure 4. Recall@K and Coverage Rate of Eclipse

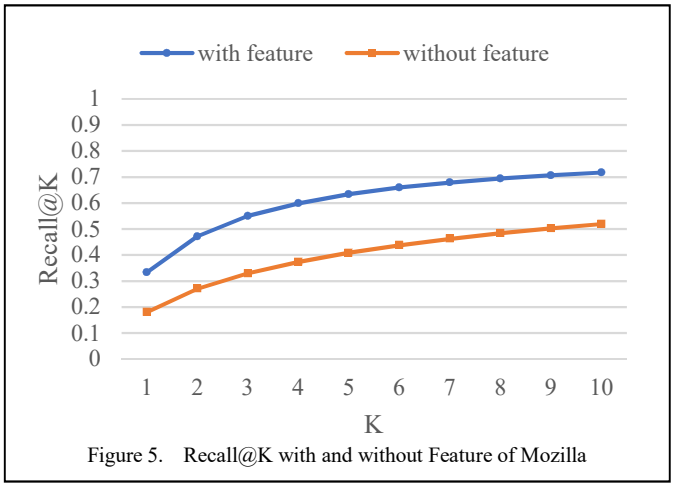


Figure 5. Recall@K with and without Feature of Mozilla

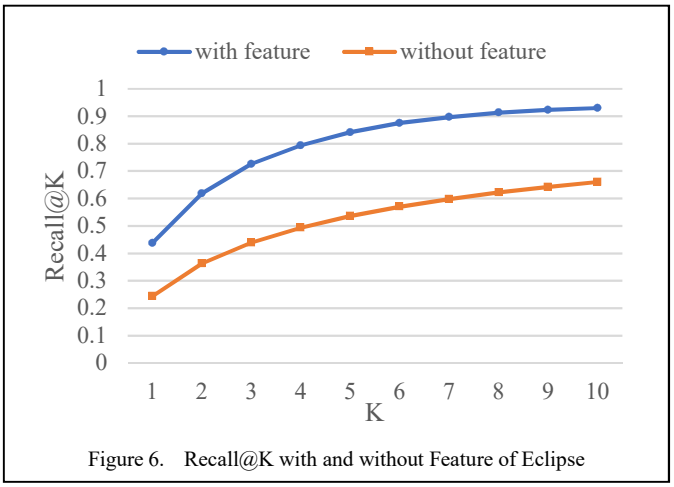


Figure 6. Recall@K with and without Feature of Eclipse

Table II presents the coverage rate of our method with and without product and component information for Eclipse and Mozilla. Note that the coverage rates are similar for both projects, which indicates our method has similar chance to pick out the actual fixer despite with or without narrowing the search scope.

Figures 5, 6 presents the values of Recall@1 to Recall@10 of our method with and without features. By narrowing the search scope with product and component information, the Recall@K values increase for both Eclipse and Mozilla. The relative improvements range from 41% to 80% for Mozilla and from 38% to 84% for Eclipse.

IV. Discussion

A. Advantages

In essence, our method is based on relevant search technique, while the machine learning based methods treat bug triage like a text classification task.

Machine learning based methods regard each developer as a class and each bug report as a document. If the number of classes is small, these methods are effective enough. However, when the number of classes increases to hundreds or even thousands, it is hard to design an accurate and efficient classifier [5].

Our method adopts relevant search technique to construct a search engine for bug reports. It utilizes the search results with the relevance scores to get a ranked recommendation list. Besides, to reduce the noise, it uses the component and product information of bug reports to narrow the search scope, which significantly improves the performance. Because our method only need add new bug reports to index for updating, and searching in the index is fast. It is easily extensible and able to deal with large-scale data using our method.

B. Limitations

Our method essentially has some limitations. In some cases, it will fail to recommend the actual developer due to the practice in bug triage.

First, if a bug is fixed by a new developer within its search scope, our method cannot recommend the right developer. Because the fixer does not correspond to any bug in the search scope, our method only recommends developers who have fixed at least one bug before. Besides, some developers may have fixed many bugs relevant to a bug. However, at the time of bug triage, they are not free for some reasons, so the bug is assigned to another less relevant developer [5].

C. Threats to Validity

Internal validity is an estimate of the degree to which a causal conclusion based on a study can be made. To improve the internal validity, we preprocess our datasets following previous studies [9] [19]. We only collected bug reports with status CLOSED and FIXED to ensure the bug reports are related to bugs and have been fixed. To determine the fixer of a bug, we examine the "assigned to" field of the bug report. However, other studies chose another way to determine fixers. They proposed a heuristic approach to extracted the fixers [1] [2]. We do not adopt their heuristics because recent studies do not follow them. Besides, we exclude the bug reports whose "assigned to" fields are generic names or appear less than 10 times to reduce the noise.

External validity is an estimate of the degree to which our experiment results can be generalized. Although we have analyzed 91251 and 100964 bug reports from Eclipse and Mozilla, our method may not appropriate to other projects. To improve external validity, we plan to experiment our method on more new bug reports from more projects in the future.

Construct validity is an estimate of the degree to which metrics can be trusted. To determine the effectiveness of our method, we need check the actual fixer is whether in the recommendation list or not. Recall@K is a proper metric for this purpose. It also has been adopted by many prior bug triage studies [5] [19].

V. Related Work

According to the different techniques used to automatic bug triage, Tao Zhang et al. [4] classify the prior studies into five categories: machine learning-based approach, expertise model-based approach, tossing graph-based approach, social network-based approach and topic model-based approach.

As a pioneering study, Cubranic et al. [1] regard bug triage as a text categorization problem. In their work, each assignee was considered to be a single class, and each bug report was assigned to only one class. Anvik et al. [2][12] experimented several different machine-learning algorithms including Naïve Bayes, SVM, C4.5 to recommend a list of appropriate developers for fixing a new bug. The experiment results show SVM performed better than others on their datasets. Ahsan et al. [15] use feature selection and Latent Sematic Indexing [16] to reduce the dimensionality of the term-to-document matrix. Their results showed the bug triage system combined LSI and SVM has the best performance. Xuan et al. [13] proposed a semi-supervised text classification method, which utilizing expectation-maximization based on both labeled and unlabeled data to enhance Naïve Bayes classifier. To remove the noisy data, Zou et al. [18] adopted feature selection. Their results showed that feature selection could improve the performance using Naïve Bayes by up to 5%. Xia et al. [14] proposed a method using a composite model called DevRec, which combined bug reports-based and developers-based analysis.

Jeong et al. [3] proposed a tossing graph model to capture bug tossing history and use this model to improve bug triaging prediction accuracy. Their experiments demonstrated the tossing graph model could improve the accuracy of automatic bug triage using machine-learning algorithms such as Naïve Bayes and Bayesian Network [17] only. Bhattacharya et al. [6][7] improved the accuracy of bug triage by utilizing a multi-feature tossing graph and increment learning.

Matter et al.[8] modeled developers' expertise using the vocabularies found in their source code files and compare them to the terms appearing in corresponding bug reports. Tamrawi et al. [9] proposed Bugzie, an automatic bug triaging tool based on fuzzy set and cache-based modeling of the bug-fixing expertise of developers. Xuan et al. [10] achieved developer prioritization via social network analysis to improve the performance of

automatic bug triage using SVM or Naïve Bayes only. Hao Hu et al. [5] utilized the historical bug-fix information to constructed a network to captures the knowledge of "who fixed what, where". They use the network to recommend suitable developers.

Naguib et al. [11] adopted LDA to cluster the bug reports into topics. Then they created the activity profile for each developer of the bug tracking repository by mining history logs and bug report topic models. An activity profile consists of two parts, including developer's role and developer's topic associations. By utilizing activity profile and a new bug's topic model., they proposed a ranking algorithm to find the most appropriate developer to fix the given bug. Xin Xia et al. [19] proposed a specialized topic modeling named MTM which extends LDA by considering product and component of information of bug reports. They used MTM to get the topic distribution of a new bug report to assign an appropriate fixer based on the affinity of the fixer to the topics.

Regarding relevant search, Zhou et al. [20] utilized indexing and searching to recommend tags for software information sites.

VI. CONCLUSION

Bug triage is a time-consuming process if all bugs are manually assigned to developers in large open source projects. In this paper, we proposed a method based on relevant search technique. To improve efficiency and effectiveness of bug triage, our method recommends developers for bugs to be assigned. Besides, we utilized component and product information of bug reports to improve the performance of our method further. We have evaluated our method on bug reports of Eclipse and Mozilla. For Eclipse, our method achieved 43.8% and 84.1% accuracies when recommending 1 and 5 developers. For Mozilla, the accuracies are 33.3% and 63.4% for recommending 1 and 5 developers. All of the results of our method outperforms the SVM and Naïve Bayes method.

With the advent of big data era [21][22], in future work, we plan to incorporate bug tossing graphs into our method to introduce more relevant developers in the top of recommendation. Also, we plan to utilize more features of bug reports to improve the performance further.

ACKNOWLEDGEMENT

This work is partly supported by National Natural Science Foundation of China (NSFC) (grant No. 61572374, U163620068, U1135005), the Academic Team Building Plan from Wuhan University and National Science Foundation (NSF) (grant No. DGE-1522883).

REFERENCES

[1] Murphy G, Cubranic D. Automatic bug triage using text categorization. InProceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering 2004.

[2] Anvik J, Hiew L, Murphy GC. Who should fix this bug?. InProceedings of the 28th international conference on Software engineering 2006 May 28 (pp. 361-370). ACM.

[3] Jeong G, Kim S, Zimmermann T. Improving bug triage with bug tossing graphs. InProceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering 2009 Aug 24 (pp. 111-120). ACM.

[4] Zhang T, Jiang H, Luo X, Chan AT. A Literature Review of Research in Bug Resolution: Tasks, Challenges and Future Directions. The Computer Journal. 2016 May 1;59(5):741-73.

[5] Hu H, Zhang H, Xuan J, Sun W. Effective bug triage based on historical bug-fix information. InSoftware Reliability Engineering (ISSRE), 2014 IEEE 25th International Symposium on 2014 Nov 3 (pp. 122-132). IEEE.

[6] Bhattacharya P, Neamtiu I. Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging. InSoftware Maintenance (ICSM), 2010 IEEE International Conference on 2010 Sep 12 (pp. 1-10). IEEE.

[7] Bhattacharya P, Neamtiu I, Shelton CR. Automated, highly-accurate, bug assignment using machine learning and tossing graphs. Journal of Systems and Software. 2012 Oct 31;85(10):2275-92.

[8] Matter D, Kuhn A, Nierstrasz O. Assigning bug reports using a vocabulary-based expertise model of developers. InMining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on 2009 May 16 (pp. 131-140). IEEE.

[9] Tamrawi A, Nguyen TT, Al-Kofahi JM, Nguyen TN. Fuzzy set and cache-based approach for bug triaging. InProceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering 2011 Sep 5 (pp. 365-375). ACM.

[10] Xuan J, Jiang H, Ren Z, Zou W. Developer prioritization in bug repositories. InSoftware Engineering (ICSE), 2012 34th International Conference on 2012 Jun 2 (pp. 25-35). IEEE.

[11] Naguib H, Narayan N, Brügge B, Helal D. Bug report assignee recommendation using activity profiles. InMining Software Repositories (MSR), 2013 10th IEEE Working Conference on 2013 May 18 (pp. 22-30). IEEE.

[12] Anvik J, Murphy GC. Reducing the effort of bug report triage: Recommenders for development-oriented decisions. ACM Transactions on Software Engineering and Methodology (TOSEM). 2011 Aug 1;20(3):10.

[13] Xuan J, Jiang H, Ren Z, Yan J, Luo Z. Automatic Bug Triage using Semi-Supervised Text Classification. InSEKE 2010 Jul (pp. 209-214).

[14] Xia X, Lo D, Wang X, Zhou B. Accurate developer recommendation for bug resolution. InReverse engineering (WCRE), 2013 20th working conference on 2013 Oct 14 (pp. 72-81). IEEE.

[15] Ahsan SN, Ferzund J, Wotawa F. Automatic software bug triage system (bts) based on latent semantic indexing and support vector machine. InSoftware Engineering Advances, 2009. ICSEA'09. Fourth International Conference on 2009 Sep 20 (pp. 216-221). IEEE.

[16] Hofmann T. Probabilistic latent semantic indexing. InProceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval 1999 Aug 1 (pp. 50-57). ACM.

[17] Friedman N, Nachman I, Peér D. Learning bayesian network structure from massive datasets: the «sparse candidate «algorithm. InProceedings of the Fifteenth conference on Uncertainty in artificial intelligence 1999 Jul 30 (pp. 206-215). Morgan Kaufmann Publishers Inc..

[18] Zou W, Hu Y, Xuan J, Jiang H. Towards training set reduction for bug triage. InComputer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual 2011 Jul 18 (pp. 576-581). IEEE.

[19] Xin Xia, David Lo, Ying Ding, Jafar M. Al-Kofahi, Tien N. Nguyen, and Xinyu Wang. Improving Automated Bug Triaging with Specialized Topic Model. IEEE Transactions on Software Engineering (TSE), IEEE CS, 2016. in press

[20] Zhou P, Liu J, Yang Z, Zhou G. Scalable Tag Recommendation for Software Information Sites. InThe 24th IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER) 2017 Feb 20.

[21] Xu, Z., Liu, Y., Mei, L., Hu, C. and Chen, L., 2015. Semantic based representing and organizing surveillance big data using video structural description technology. Journal of Systems and Software, 102, pp.217-225.

[22] Liu, J., Yu, X., Xu, Z., Choo, K.K.R., Hong, L. and Cui, X., 2016. A cloud‐based taxi trace mining framework for smart city. Software: Practice and Experience.

Concurrent Call Level Interfaces

Based on an Embedded Thread Safe Local Memory Structure

Óscar Mortágua Pereira
Instituto de Telecomunicações
DETI – University of Aveiro
Aveiro, Portugal
omp@ua.pt

Rui L. Aguiar
Instituto de Telecomunicações
DETI – University of Aveiro
Aveiro, Portugal
ruilaa@ua.pt

***Abstract*— Performance is traditionally considered one of the most significant concerns in intensive database applications. Several architectural tactics may be taken to minimize the possibility of coming across with any performance bottleneck. One of them is the usage of Call Level Interfaces (CLI). CLI are low level API that provide a high performance environment to execute SQL statements on relational and also on some NoSQL database servers. In spite of this, CLI are not thread safe, this way preventing distinct threads from sharing datasets retrieved from databases through Select statements. Thus, in situations where two or more threads need to access datasets retrieved from the same Select statement, there is no other alternative than providing each thread with its own dataset, this way consuming important computational resources. In this paper we propose a new design for CLI to overcome the aforementioned drawback. Unlike current implementations of CLI, now they are natively thread-safe. The implementation herein presented is based on a thread safe updatable local memory structure where data retrieved from databases is kept. A proof of concept based on Java Database Connectivity type 4 (JDBC) for SQL Server 2008 is presented and also a performance assessment.**

Keywords— call level interfaces, concurrency, performance, databases, middleware, software architecture.

I. INTRODUCTION

Database applications comprise at least two main components: database components and application components. In our context, application components are developed in the object-oriented paradigm and database components rely on the relational paradigm. The two paradigms are simply too different to be bridged seamlessly leading to difficulties informally known as impedance mismatch [1]. The diverse foundations of both paradigms are a major hindrance for their integration, being an open challenge for more than 50 years [2]. These challenges are especially noticeable in environments where code production is under strict development deadlines and where code development efficiency is a major concern. In order to overcome the impedance mismatch issue, several solutions have emerged [1][2][3][4][5][6]. Despite their individual advantages, these solutions have not been developed to address situations where users need to implement concurrent mechanisms over the in-memory data structures returned by Select statements. As generally accepted, performance is one of the most challenging non-functional software requirements in database applications. Here, Call Level Interfaces (CLI) have to be considered as a promising alternative [2]. CLI are programming API aimed at easing the integration of client software components with database components. They use native SQL statements, this way promoting the SQL expressiveness and the SQL performance. Nevertheless, CLI do not provide some of the most well-known and common features to improve system performance, being concurrency the most paradigmatic case. We cannot forget that the speed of data creation and data storage increases every passing day, which is followed by an increased need of power computation to process it. Very often, part of the increased need of power computation derives from the incapacity of current systems to share resources that have already been become available. For example, let us consider a Select statement that retrieves a dataset from a database, which is kept in local memory structures (LMS): ResultSet [7] and RecordSet [8] are examples of LMS for JDBC and ODBC, respectively. Probably, the data contained in LMS could be made available and shared concurrently to several consumers (threads). This possibility would eliminate the need to retrieve and duplicate the same datasets over and over again for each thread (in different LMS instances). To achieve this goal, the access to LMS needs to be thread-safe to avoid unwanted conflicts. Unfortunately, current tools used to develop business logics are not thread-safe. They were mainly designed to minimize the impedance mismatch between the object-oriented and the relational paradigms. Among those options, CLI are considered the best option whenever performance is a non-functional key requirement [9]. For this reason, CLI were chosen to design a thread-safe API to be used on the building process of business logics. The proposal herein presented is based on a modification on the native internal LMS structure in order to make it natively thread-safe. This means that CLI are now implemented with embedded concurrent mechanisms, in opposite to [10][11]. A proof of concept based on Java Database Connectivity type 4 (JDBC) is presented. A performance assessment is also conducted in order to evaluate the performance of both architectures.

The remainder of this paper is organized as follows: chapter II presents the motivation for this research, chapter III describes the current state of the art, chapter IV presents the required background to keep this paper as self-contained

DOI reference number: 10.18293/SEKE2017-120

as possible, chapter V presents the proposal for thread-safe CLI, chapter VI presents a performance assessment and, finally, chapter VII presents the final conclusion.

II. MOTIVATION

In this section we present the limitations of current CLI and the goal we want to achieve. The presentation is based on simple examples to avoid any discomfort of readers less knowledgeable about CLI.

The scenario to be addressed comprises situations where there is the need to concurrently share access to datasets retrieved by Select statements. These datasets are managed by LMS. The access to data contained by LMS is row and attribute oriented. This means that at any given moment just one row may be selected and, then, the access to data is processed by selecting one attribute at time. After being processed, another row may be selected and the process continues. If no other control is implemented, this context is not compatible with multi-thread environments. Listing 1 depicts a situation where two threads are using the same LMS instance (lms). Regarding thread A, it scrolls to the next row and then it reads the first attribute. Thread B moves to row number 10 and then it reads the third attribute. In preemptive multitasking [12] environments, these two threads may enter in a conflict state. Suppose that Thread A is the running and it scrolls to line 5. If Thread B becomes the running thread, it will move to row 10, it reads the attribute number 3 and then voluntarily it suspends itself. When thread A is resumed, it will read the attribute number 1, not from row 5 as initially expected but wrongly from row 10. CLI do not provide any feature to prevent this from happening. To overcome this situation, an initial approach has been already proposed to overcome this CLI drawback, which it is based on a wrapper that hides the functionalities of CLI and exposes thread safe services [10][11].

```
// Thread A
1 lms.moveNextRow();
2 id=lms.read(1);
3 …
```

```
// Thread B
1 lms.moveToRow(10);
2 name=lms.read(3);
3 thread.suspend();
4 …
```

Listing 1. Two threads accessing the same LMS.

III. STATE OF THE ART

A survey has been carried out around tools aimed at integrating client applications and databases. The survey comprises the most popular tools, such as Hibernate [4], Spring [13], TopLink [14], JPA [5] and LINQ [15]. These tools may provide concurrency but always at a very high level. Basically, they provide some locking policies to synchronize read and write actions. But these read and write synchronized actions are not executed over the same memory location. They are executed over distinct objects, such as sessions in Hibernate. These objects (sessions) are not thread-safe and therefore do not provide any protocol to access concurrently the in-memory data contained on LMS.

A survey has also been carried out about two approaches proposed by the research community: SQL DOM [16] and Safe Query Objects [17]. SQL DOM generates a Dynamic Link Library containing classes that are strongly-typed to a database schema. These classes are used to construct dynamic SQL statements without manipulating any strings. Safe Query Objects combine object-relational mapping with object-oriented languages to specify queries using strongly-typed objects and methods. They rely on Java Data Objects to provide strongly-typed objects and also to provide data persistence. These proposals are focused on minimizing the impedance mismatch. None of these approaches address concurrency at any level.

In [18] a different approach is presented to address the lack of concurrent mechanisms of CLI. Concurrency is implemented by an explicit locking mechanism based on two methods: *lock()* and *unlock*. Programmers are responsible for invoking these methods correctly in order to control the exclusive access mode to LMS. Additionally, the conducted assessment is based on a fixed number of rows which does not convey a dynamic perspective of the performance for different scenarios.

Aspect-oriented programming [19] community considers persistence as a crosscutting concern [20]. Several works have been presented but none addresses the points here under consideration. The following works are emphasized: [21] is focused on separating scattered and tangled code in advanced transaction management; [20] addresses persistence relying on AspectJ; [22] presents AO4Sql as an aspect-oriented extension for SQL aimed at addressing logging, profiling and runtime schema evolution. It would be interesting to see an aspect-oriented approach for the points herein under discussion.

The research presented in [11][10] proposes an architecture based on a wrapper which hides the CLI functionalities and exposes thread-safe services. It is known as CTSA – Concurrent Tuple Set Architecture. This approach is clearly an improvement when compared with the one presented in [18] but the thread-safe mechanisms are not embedded on CLI as proposed in this research. Nevertheless, that approach will also be used to compare their results with the results obtained by the approach herein proposed. As it will be shown, the herein presented approach clearly improves the performance achieved with CTSA.

In this master thesis [23] an identical approach, as the one herein presented, has been designed but the final results were not convincing. This paper presents a new implementation.

To the best of our knowledge no other researches have been conducted around concurrency on LMS of CLI.

IV. BACKGROUND

In this section we present the necessary background to make this a self-contained paper. It is divided in two main sub-sections. In the first one, some fundamental functionality of LMS are provided and in the second one a brief description is given how CLI and Relational Database Management Systems (RDBMS) interact with each other.

A. Functionality of LMS

LMS are client-side object-oriented abstractions of a

relational concept: the server side cursor. LMS are instantiated to manage relations returned by Select expressions. At instantiation time, some runtime properties of LMS are defined to characterize their functionalities. Two main groups of functionalities are herein emphasized: scrolling functionalities and accessing functionalities (they are orthogonal). Scrolling functionalities provide two main types of LMS (they are mutual-exclusive): forward-only – rows are read sequentially from the first one till the last one, and scrollable – rows can be randomly read. Accessing functionalities provide two main types of LMS (they are mutual-exclusive): read-only and updatable LMS. While read-only LMS only provide one protocol to read their contents, updatable LMS, beyond the read protocol, also provide three additional protocols: update (to update their contents), insert (to insert new rows) and delete (to delete existing rows). Another relevant issue is the mechanism implemented for each protocol (read, update, insert and delete). LMS are row oriented and protocol oriented. This has two main implications. First, at any time only one row can be selected as the target row. Second, if an update or insert protocol is being executed, applications cannot start any another protocol. If this rule is not fulfilled, LMS discard changes made during the previous protocol. Table 1 concisely presents how the 4 main LMS protocols work: 1 – read protocol, 2 – update protocol, 3 – insert protocol and 4 – delete protocol.

Table 1. 4 main protocols of LMS.

1	Point to a row Read attributes Point to another row	2	Point to a row Update attributes Commit update
3	Start insert Insert attributes Commit insert	4	Point to a row Delete row

B. *Interaction Between CLI and RDBMS*

The communication between CLI and RDBMS relies on proprietary protocols of RDBMS vendors but their general interaction follows the structure presented in Figure 1. When a Select expression is executed, RDBMS create a server dataset with the retrieved data and also a server cursor. All or only a part of the retrieved data is copied from server datasets to LMS depending on the LMS properties. When data is partially transferred to LMS, new blocks of data are transferred whenever client applications need to access data not contained locally in the LMS. The relationship between LMS and cursors, and between cursors and LMS are all 1 to 1. This means that whenever a Select statement is executed, there will be one additional LMS and one additional server cursor. We emphasize that one additional means resources (LMS, cursors and datasets) that are being replicated, very probably unnecessarily. In scenarios where there are simultaneously several server datasets from the same Select statement, we are before a situation where there is an unnecessarily wasting of computational resources.

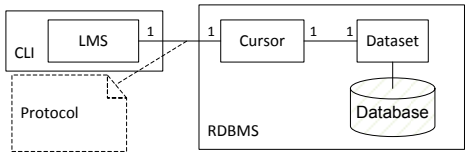


Figure 1. Connection between LMS and RDBMS.

V. PROPOSED ARCHITECTURE FOR CONCURRENT CLI

In this section we present the architecture that has been defined to design concurrent LMS. To achieve the proposed goal, some source code of CLI was redesigned. In this research, in opposite to CTSA, we explored the usage of embedded thread-safe LMS on which threads interact directly with the data retrieved from databases. To achieve this goal, several CLI interfaces (services) need to be rewritten, such as those aimed at: scrolling on LMS, reading data from LMS, updating data on LMS, inserting data on LMS and, finally, deleting data on LMS. These are the fundamental interfaces responsible for providing services through which client applications are currently able to interact with LMS. A new concurrent component, known as CLMS (Concurrent LMS), replaces the default LMS. Figure 2 presents the main architecture of CLMS which contains a local cache to store the retrieved data. Basically it implements a general interface (ICLMS) which extends the 6 fundamental additional interfaces: ICForwardOnly, ICScrollable (forward-only and scrollable CLMS, respectively) and ICRead, ICUpdate, ICInsert and ICDelete (read update, insert and delete protocols, respectively). Please remember that these are the basic services required to interact with CLMS, as previously mentioned in chapter IV.

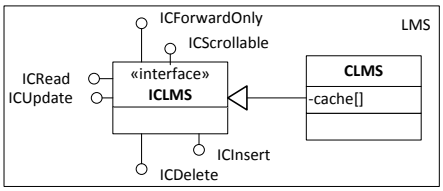


Figure 2. Concurrent architecture for CLMS.

Before, presenting some additional details, the concept of execution context is introduced. Each running thread has its own execution context which consists on the protocol being executed (if any) and the current selected row. This is an important concept because it is based on it that it was possible to design a thread-safe LMS. Basically, every time a thread enters the monitor (thread-safe area) it needs to set its execution context and before leaving the monitor it must store its execution context. This process will ensure that each thread, whenever initiating the access to the monitor, it is able to restore its previous execution context (protocol being executed and row being accessed). Now, we can introduce how an exclusive access mode can be implemented to access CLMS. Two methodologies are proposed: method oriented access mode and protocol oriented access mode. Basically, the method oriented access mode requires a restoring and storing process for the execution context every time a method is executed on CLMS, while the protocol oriented access mode does only require a restoring and storing process by each protocol being executed. Let us take a closer look to the protocols to evaluate the options that are available for each one. The scrolling protocol involves one method at a

time and, therefore, the obvious approach is the method oriented access mode. Access modes for Insert, Update and Delete protocols do not have any other alternative but being implemented as protocol oriented access mode. As mentioned before, this derives from the fact that these protocols, while being executed, cannot be preempted to start any other different protocol. Read protocol may be implemented in any access mode protocol: method access mode (operating on an attribute by attribute basis) or protocol access mode (operating on a row by row basis). In order to implement the exclusive access mode to CLMS it was decided, based on practical evidence and empirical experience, to use method oriented access mode for the ICForwardOnly and ICScrollable interfaces and protocol oriented access mode for the remaining interfaces. We assume that in the most common situations, several attributes are read in each Read protocol, this way not advising the method oriented access mode. With the thread-safe LMS the resort to multiple cursors in the database server is avoided. A single server cursor is able to satisfy simultaneously several client side threads sharing the same LMS.

VI. PERFORMANCE ASSESSMENT

A performance assessment was carried out to compare a solution based on a traditional non-shared (S-JDBC) LMS and the one herein proposed (C-JDBC). The performance assessment ran in a context completely identical to one used in CTSA, this way ensuring that the collected results can be compared to evaluate the impact of thread-safe LMS.

C-JDBC uses a unique LMS that is shared by all threads, while in S-JDBC each thread has its own LMS. All LMS contain the same relation returned by the same Select expression. Concisely, Figure 3, presents the block diagram for the used scenario during the assessment process.

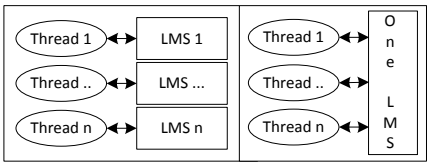


Figure 3. Left side: S-JDBC, right side: C-JDBC.

Three scenarios were defined for the main operations of both solutions: Select (*s*), Update (*u*) and Insert (*i*). Each scenario comprises a set of several numbers of rows to be processed *[nr]* for both S-JDBC and C-JDBC, and a set of several numbers of simultaneous running threads *[nt]* for both S-JDBC and C-JDBC. In order to formalize the entities' representation the following definition is presented: $E_{(\alpha,\gamma)}$ *([nt], [nr])* where $\alpha \in$ {c-jdbc, s-jdbc}, and $\gamma \in \{s,u,i\}$. To simplify the general formalization, $E_{(\alpha,\gamma)}$ *([nt], [nr])* is represented by default as $E_{(\alpha,\gamma)}$. Each scenario comprises a specific goal which is known as a *task*. A task represents a particular case for the use of C-JDBC and S-JDBC. The tasks to be performed are: Read (read *[nr]* adjacent tuples from the LMS), Update (update *[nr]* adjacent rows of a LMS) and Insert (insert *[nr]* tuples into a LMS). Please remember that S-JDBC uses *[nt]* LMS while C-JDBC always uses one LMS. The assessment could also comprise a random access pattern but, by empirical evidence, the most common access pattern is the access to adjacent rows. It was also decided to create and enforce different contexts for S-JDBC and C-JDBC. The idea is to create a context to C-JDBC based on tactics aimed at decreasing its performance while executing the defined tasks while for S-JDBC we will use tactics to enforce the opposite effect. S-JDBC favorable tactics – each thread has its own LMS and rows are always sequentially selected in order to minimize the transference of rows between JDBC and SQL Server. C-JDBC unfavorable environment - two situations were implemented: 1) Each thread will auto-suspend itself after having executed one protocol: read one row, update one row or insert one row. This will give the opportunity to other thread to become the running thread, this way maximizing the number of changes in the execution contexts. 2) All threads share the LMS but each thread has its own adjacent rows. This means that when a thread becomes the running thread, its execution context will set a row that belongs to a different set of rows, this way maximizing the number of blocks to be transferred from SQL Server.

Table 2 presents the algorithms used to assess S-JDBC and C-JDBC. All scenarios, for each solution, share the same algorithm for the assessments to be carried out. C-JDBC and S-JDBC create the same number of threads (*nt*) and each thread processes the same number of rows (*nr*). The main difference is: while in S-JDBC each thread selects its own subset of rows, this way accessing its own LMS, in C-JDBC a LMS is shared by all threads containing all rows.

The test-bed comprises two computers: PC1 - Dell Latitude E5500, Intel Duo Core P8600 @2.40GHz, 4.00 GB RAM, Windows Vista Enterprise Service Pack 2 (32bits), Java SE 6, JDBC(sqljdbc4); PC2 – Asus-P5K-VM, Intel Duo Core E6550 @2,33 GHz, 4.00 GB RAM, Windows XP Professional Service Pack 3, SQL Server 2008. C-JDBC is executed in PC1 and SQL Server runs in PC2. In order to promote an ideal environment the following actions were taken: CPU were set to run with a single core, this way maximizing the influence of the implemented solutions; the running threads were given the highest priority; all non-essential processes/services were cancelled in both PCs and a direct and dedicated network cable connecting PC1 and PC2 has been used in exclusive mode and performing 100MBits of bandwidth. In order to avoid any overhead added by SQL Server, some default SQL Server database properties were changed as, Auto Update Statistics = false and Recovery Model = Simple.

The sets used for the number of rows and for the number of threads were:

[nt]={1, 5, 10, 25, 50, 75, 100, 150, 200, 250, 350, 500}

[nr]={5, 10, 25, 50, 75, 100}

25 raw measures were collected for each $E_{(\alpha,\gamma)}$*([nt],[nr])* leading to (2x3x12x6)x25=10,800 raw measures. Intermediate measures were computed from the average of the 5 best measures of each $E_{(\alpha,\gamma)}$*([nt],[nr])* leading to a total of 2x3x12x6=432 measures. The final measures used in the next charts represent the ratios between $E_{(c\text{-}jdbc,\gamma)}$ and $E_{(s\text{-}}$

$_{jdbc,\gamma)}$ for each *([nt],[nr])*. In all charts the vertical axis is for the ratios and the horizontal axis is for the *[nt]*.

A table Student with the following schema was also created to store the data being used: id (int, pk), firstName (varchar 25), lastName (varchar 25), crdId (int), regYear (int), applGrade (float).

Table 2. Algorithms for $E_{(c\text{-}jdbc,\gamma)}$ II-Algorithms for $E_{(s\text{-}jdbc,\gamma)}$.

I	1. Delete all rows from *Student* 2. Fill *Student* with *[nr]*[nt]* rows (zero rows for insert) 3. Start counter 4. Select all rows from *Student* into one single ResultSet 5. Create all threads. Each thread (ψ tuples) 5.1 for each row 5.1.1 read/update/insert (row) 5.1.2 suspend thread 5.2 dies 6. Wait all threads to die 7. Stop counter
II	1. Delete all rows from *Student* 2. Fill *Student* with *[nr]*[nt]* rows (zero rows for insert) 3. Start counter 4. Create all threads. Each thread: 4.1 select ψ trow into its own ResultSet 4.2 for each row 4.2.1 read/update/insert a tuple 4.3 dies 5. Wait all threads to die 6. Stop counter

Select scenario

Figure 4 presents the ratio between the measures collected for $E_{(s\text{-}jdbc,s)}$) and $E_{(c\text{-}jdbc,s)}$. The chart shows that there are some situations where the gain is very significant. The most significant situation occurs for 5 tuples and 10 to 25 threads reaching a gain above 3.5 times. The ratio decreases when the number of rows increases and also when the number of threads increases. The reasons for this behavior is that when either *nt* or *nr* increases, the probability of a thread of C-JDBC to access a row not contained in the LMS increases, this way requiring a block transfer from the server dataset to the LMS. Please remember that in C-JDBC all threads share the same LMS and each thread is reading a different block of adjacent rows. This means that when *thread 1* reads row 1, *thread n* is reading row *nr*(n-1)* which eventually may not be at that moment contained in the local LMS.

Figure 5 presents a tabular view of the chart presented in

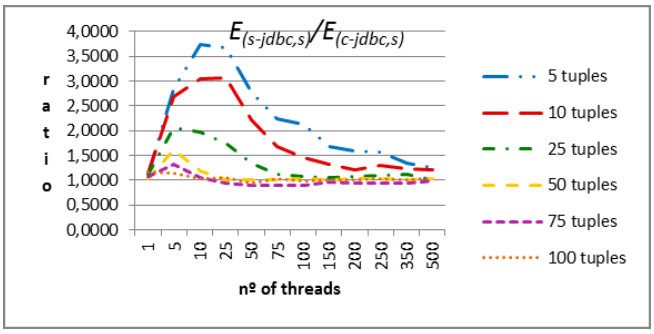


Figure 4. Select scenario: ratio between S-JDBC and C-JDBC.

NR/NT	5	10	25	50	75	100
1	1,13	1,11	1,13	1,13	1,09	1,14
5	2,68	2,47	1,99	1,50	1,32	1,16
10	3,54	2,69	1,63	1,17	1,01	1,04
25	3,48	2,73	1,59	1,02	0,96	1,04
50	2,61	1,67	1,21	0,90	0,91	0,97
75	1,76	1,26	0,92	0,94	0,91	0,97
100	1,96	1,42	0,87	0,95	0,93	0,98
150	1,38	1,04	0,95	0,94	0,96	0,98
200	1,37	1,19	0,97	0,97	0,95	1,00
250	1,42	1,10	1,01	0,95	0,94	0,99
350	1,14	1,04	1,03	0,93	0,98	1,03
500	1,23	1,00	1,02	0,94	0,94	1,03

Figure 5. Tabular view for the ratio between S-JDBC and C-JDBC for the Select scenario.

Figure 4. It shows that in some cases the ratio is not greater than one which means that measures of $E_{(c\text{-}jdbc,s)}$ are not always better than measures of $E_{(s\text{-}jdbc,s)}$. But the key issue is that when compared with ratios obtained with CTSA, C-JDBC has a mean improvement around 6%. Additionally, in this assessment only about 25 ratios are under 1.00, in opposite to 40 in CTSA. This improvement is due to the embedded thread-safe mechanisms. CTSA was based on a wrapper and, therefore, an overhead is an unavoidable issue which was solved in the approach herein presented. The results here obtained show that $E_{(c\text{-}jdbc,s)}$ has achieved outstanding results even when compared with the CTSA.

Update and Insert scenarios

The update scenario updates rows contained by LMS and the insert scenario inserts rows in empty LMS. The measures collected for the Update and also for the Insert scenario were very close to ones collected for CTSA. The basic reason for these results is that these protocols are much heavier than the Select protocol and, therefore, the achieved gains, in C-JDBC when compared with CTSA, have a much lower impact. Figure 6 presents the graphic for the Update scenario and Figure 7 presents the graphic for the Insert scenario for the ratios S-JDBC/C-JDBC.

The chart for Update scenario shows that the gain is always greater than 1 and it increases when the number of rows decreases. The number of threads seems to not have a significant impact. This behavior is understandable if we remind that the update protocol is very heavy and its weight can be much more influent than the weight associated with

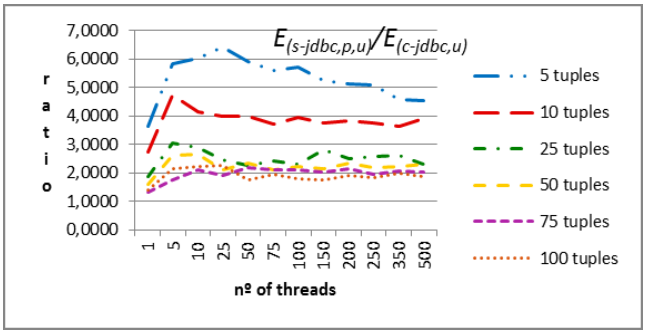


Figure 6. Update scenario: ratio between S-JDBC and C-JDBC.

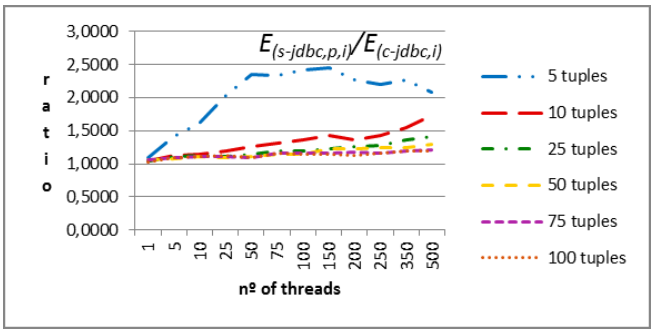


Figure 7. Insert scenario: ratio between S-JDBC and C-JDBC.

transferring blocks from dataset servers to LMS. The maximum gain (above 6) is reached for 5 tuples and 25 threads. The chart for the Insert scenario shows that the gain is always greater than 1 and that it increases when the number of tuples decreases and also when the number of threads increases. This last behavior is curious. It derives from the fact that server datasets are empty and there are no transferences of blocks from dataset servers to LMS. This way, as the number of server datasets increases, performance of S-JDBC decreases more significantly than performance of C-JDBC.

VII. CONCLUSION

Performance has become increasingly a key concern in intensive database applications. Recently is has reached a major importance with the advent of Big Data and IoT. Among many issues that can influence the overall performance, the middleware that connects business logics to RDBMS and NoSQL servers is certainly a key component, in our case, CLI. CLI are composed by two main types of components: client-side components and server side components. These components were designed to perform in environments where concurrency is not a major concern. Basically, both types of components are not prepared to work on client-side environments where several threads need to access to the same memory structures, especially LMS. To overcome this drawback, we propose a new design for CLI where LMS are natively thread-safe, in opposite to the work done with CTSA, The collected results show that the improvement in performance is noticeable in the Select scenario even when compared with the results collected with CTSA. The ratios have been improved in a mean of 6% and the number of ratios < 1.0 felt from 40 to 25. In the remaining scenarios, Update and Insert, the collected results are very similar. This is due to the fact that Update and Insert protocols are much heavier than the Select, leading to a much lower percentage impact in the overall performance.

As a future work, we are already working on an extension of the C-JDBC which will provide an additional functionality. Basically, besides the single thread-safe LMS already implemented, it will also provide replicas of the same LMS. This way, each thread will own its own LMS but the server will only need a single server cursor. The replication process can bring many advantages in many situations (now threads can interact with LMS without any locking mechanism) but the update and insert processes need additional processing to keep data consistency in all LMS. Anyway, the preliminary results are very encouraging.

ACKNOWLEDGEMENTS

This work is funded by National Funds through FCT - Fundação para a Ciência e a Tecnologia under the project UID/EEA/50008/2013.

REFERENCES

[1] ISO, "ISO/IEC 9075-3:2003," 2003. [Online]. Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=34134.
[2] Microsoft, "Microsoft Open Database Connectivity," 1992. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms710252(VS.85).aspx.
[3] M. Parsian, *JDBC Recipes: A Problem-Solution Approach*. NY, USA: Apress, 2005.
[4] B. Christian and K. Gavin, *Hibernate in Action*. Manning Publications Co., 2004.
[5] D. Yang, *Java Persistence with JPA*. Outskirts Press, 2010.
[6] D. Kulkarni, L. Bolognese, M. Warren, A. Hejlsberg, and K. George, "LINQ to SQL: .NET Language-Integrated Query for Relational Data." Microsoft.
[7] Oracle, "ResultSet," 2012. [Online]. Available: http://docs.oracle.com/javase/6/docs/api/java/sql/ResultSet.html.
[8] Microsoft, "RecordSet (ODBC)," *Microsoft*. [Online]. Available: http://msdn.microsoft.com/en-us/library/5sbfs6f1.aspx. [Accessed: 16-Nov-2016].
[9] W. Cook and A. Ibrahim, "Integrating programming languages and databases: what is the problem?," 2005. [Online]. Available: http://www.odbms.org/experts.aspx#article10.
[10] Ó. M. Pereira, R. Aguiar, and M. Santos, "A Concurrent Tuple Set Architecture for Call Level Interfaces," in *Computer and Information Science*, vol. 493, R. Lee, Ed. Springer International Publishing, 2013, pp. 143–158.
[11] Ó. M. Pereira, R. L. Aguiar, and M. Y. Santos, "CTSA: Concurrent Tuple Set Architecture Extending Concurrency to Call Level Interfaces," *IJSI - Int. J. Softw. Innov.*, vol. 1, no. 3, pp. 12–33, 2013.
[12] C. J. Fidge, "Real-Time Schedulability Tests for Preemptive Multitasking," *Real-Time Syst.*, vol. 14, no. 1, pp. 61–93, 1998.
[13] Spring, "Spring." [Online]. Available: http://www.springsource.org/.
[14] Oracle, "Oracle TopLink," 2011. [Online]. Available: http://www.oracle.com/technetwork/middleware/toplink/overview/index.html.
[15] M. Erik, B. Brian, and B. Gavin, "LINQ: Reconciling Object, Relations and XML in the .NET framework," in *ACM SIGMOD Intl Conf on Management of Data*, 2006, p. 706.
[16] A. M. Russell and H. K. Ingolf, "SQL DOM: compile time checking of dynamic SQL statements," in *27th Int. Conf. on Software Engineering*, 2005, pp. 88–96.
[17] R. C. William and R. Siddhartha, "Safe query objects: statically typed objects as remotely executable queries," in *27th Int. Conf. on Software Engineering*, 2005, pp. 97–106.
[18] Ó. M. Pereira, R. L. Aguiar, and M. Y. Santos, "Assessment of a Enhanced ResultSet Component for Accessing Relational Databases," in *ICSTE-Int. Conf. on Software Technology and Engineering*, 2010, p. V1:194-201.
[19] J. L. Gregor Kiczales Anurag Mendhekar, Chris Maeda, Cristina Lopes Videira, Jean-Marc Loingtier, Joh Irwin, "Aspect-Oriented Programming," in *ECOOP*, 1997, pp. 220–242.
[20] R. Laddad, *AspectJ in Action: Practical Aspect-Oriented Programming*. Greenwich,CT,USA: Manning Publications, 2003.
[21] J. Fabry and T. D'Hondt, "KALA: Kernel Aspect Language for Advanced Transactions," in *Proceedings of the 2006 ACM Symposium on Applied Computing*, 2006, pp. 1615–1620.
[22] T. Dinkelaker, "AO4SQL: Towards an Aspect-Oriented Extension for SQL," in *8th Workshop on Reflection, AOP and Meta-Data for Software Evolution (RAM-SE'11)*, 2011, pp. 1–5.
[23] D. Gomes, Ó. M. Pereira, and W. Santos, "JDBC (Java DB connectivity) concorrente," University of Aveiro, ria - institutional repository, http://hdl.handle.net/10773/7359, 2011.

Parallel Execution Optimization of GPU-aware Components in Embedded Systems

Gabriel Campeanu
Mälardalen Real-Time Research Center
Mälardalen University, Västerås, Sweden
Email: gabriel.campeanu@mdh.se

***Abstract*—Many embedded systems process huge amount of data that comes from the interaction with the environment. The Graphics Processing Unit (GPU) is a modern embedded solution that tackles the efficiency challenge when processing a lot of data. GPU may improve even more the system performance by allowing multiple activities to be executed in a parallel manner. In a complex component-based application, the challenge is to decide the components to be executed in parallel on GPU when considering different system factors (e.g., GPU memory, GPU computation power).**

In the context of component-based CPU-GPU embedded systems, we propose an automatic method that provides parallel execution schemes of components with GPU capabilities. The introduced method considers hardware (e.g., available GPU memory), software properties (e.g., required GPU memory) and communication pattern. Moreover, the method optimizes the overall system performance based on component execution times and system architecture (i.e., communication pattern). The validation uses an underwater robot example to describe the feasibility of our proposed method.

***Keywords*—CBD, component-based development, CPU-GPU, embedded systems, GPU-aware component, GPU component, parallel component execution, optimization**

I. Introduction

An embedded system is a computational systems that may be part of a larger system executing one or several specific tasks. Many of the modern embedded systems deal with huge amount of data due to e.g., interaction with the environment. For example, underwater robots use their sensors (e.g., cameras, radars) to receive information about the surrounding underwater environment and process it in order to e.g. detect various objects.

Due to the limited computing power and the sequential execution model of the CPU, traditional embedded systems have a challenge in providing the required performance level demanded by applications, while processing huge amount of data. For example, an underwater robot, while navigates, needs to identify with a certain performance the encountered objects in order to cope with the environment changes such as detecting moving objects.

One feasible solution that tackles the efficiency challenge of processing huge amount of data is the GPU. Due to its parallel execution architecture, the GPU can efficiently process data in a parallel manner. Today, using the latest technological improvements, there are embedded boards that contain GPUs also know as CPU-GPU embedded boards such as NVIDIA Jetson TK1 and AMD Kabini.

Another trend in the development of embedded systems is the usage of component-based development (CBD) [1] [2]. This software engineering methodology promotes development of systems through the composition of already existing software units known as (software) components. A key concept of CBD is the encapsulation, where components are seen as black boxes and the source code is encapsulated (inside the components). CBD is an attractive solution for embedded systems due to e.g., its faster time-to-market and the increase of the development productivity. The industry has successfully adopted CBD for embedded system development through component models such as AUTOSAR, Koala, Rubus and IEC 61131.

In a component-based CPU-GPU embedded system, the GPU, besides an increased efficiency w.r.t. data processing, may improve the system performance by allowing multiple components to be processed in parallel. Due to the fact that the GPU is characterized by a different architecture then the traditional CPU, different rational needs to be used when targeting GPU parallelism. The challenge of specifying how many components may be parallel executed relies not only on the available hardware resource description (e.g., available GPU memory) but also on the component specifications (e.g., GPU memory usage) and the system communication pattern. For example, the sum of the GPU memory usage of components that are parallel executed needs to be lower than the available hardware memory; moreover, the components should be independent of each other w.r.t. data communication.

In this work, we provide a method that automatically computes schemes of components that can be executed in parallel on GPU. The method is formally described and includes hardware and software specifications (Section V). Furthermore, the method provides optimized solutions considering component extra-functional properties (i.e., execution time). Resembling with the bin-packing problem and the multiprocessor scheduling problem which are combinatorial NP-hard problems [3], heuristics need to be utilized in finding solutions. For the implementation part, we use a mixed-integer non-linear programming (MINLP) heuristic approach to compute execution schemes (Section VII). As validation, we utilize an underwater robot example to describe the feasibility of our method (Section VIII).

DOI reference number: 10.18293/SEKE2017-137

II. CPU-GPU Embedded Systems

Initially, GPUs appeared in late 90s and were used only in graphics-based applications. Nowadays, the GPU is used in various types of applications such as cryptography [4] and simulation of bio-molecular systems [5], becoming a general-purpose unit.

The GPU may be seen as a complementary unit to the CPU. While the CPU is designed to rapidly execute in a sequential manner each instruction, the GPU, being equipped with thousand of computation threads, excels in parallel data processing. Therefore, CPU-GPU applications perform best when distributing the right job (i.e., sequential and parallel) to the right processing unit (i.e., CPU and GPU). For example, the massive amount of data of a vision system is processed onto GPU while the CPU takes care of e.g., histogram computations [6]. An important characteristic of the GPU is that it cannot function without the CPU. Considered the brain of a system, the CPU is the one that triggers all the activities that are executed on GPU. We need also to mention that, once an activity is started to be executed by GPU, it can not be paused or preempted.

GPU integration to embedded boards is today possible through various type of systems such as NVIDIA Jetson TK1, AMD Kabini and ARM MALI. The CPU-GPU embedded systems may increase the system performance of existing applications. For example, the stereo matching application for embedded systems has a considerable increase of frame rate processing when is performed onto GPU [7].

Another trend in embedded systems is the usage of component-based development. Through the existing component models, CBD is successfully used in industry; we mention AUTOSAR that became a standard in automotive development, Koala used by Philips and IEC 61131 used for programmable logic controllers. Another industrial component model is Rubus used by e.g., Volvo Construction Equipment branch [8]. Rubus follows the pipe-and-filter interaction style that provides a precise control flow of the system which makes Rubus applicable to particular embedded system domains (e.g., real-time systems and safety systems) [9].

Following the CBD approach, a CPU-GPU system is constructed from: *i)* regular components with CPU characteristics, implemented to be executed onto CPU; and *ii)* components with GPU functionality[1] (i.e., GPU-aware components) that have both CPU and GPU characteristics. A GPU-aware component has its functionality (or part of it) specific developed to be executed on GPU. Besides the functional description, a GPU-aware component is characterized by qualities and constraints known as extra-functional properties (EFPs). For example, a GPU-aware component may have as quality the execution time performance while as constraints, it may demand a specific number of GPU threads and memory usage.

The development of GPU-based applications is realized through different programming models and the two most known are CUDA and OpenCL. While CUDA is developed by the NVIDIA vendor to be used only for their GPUs, OpenCL is a general model developed by KHRONOS group that targets GPUs produced by e.g., INTEL, AMD and NVIDIA.

[1]During the rest of the paper, we refer to a component with GPU functionality as a GPU-aware component

III. Problem Description

For traditional CPU-based embedded systems, a way to achieve parallelism is to assign components to different e.g., CPU cores, and let the components to be executed in parallel (by the OS). We assume that there are enough resources and components are data independent of each other (i.e., do not have communication connections). Basically, the parallelism is influenced by the number of distinct processing units that a system is composed of. For example, an embedded system with a quad-core CPU can have at a given time instance a maximum of four components that can be executed in parallel.

Due to a different architecture, the way to achieve parallelism on GPU is different than on CPU. A GPU is composed of hundreds of computation cores and thousands of threads and when an activity is executed, it may consume tens of cores and thousands of threads. Therefore, the metric to reason about GPU parallelism is not the numbers of cores but is a relation between several factors such as the number of independent software activities, hardware limitation (e.g., available GPU memory) and resources demands (e.g., computation threads usage). Each GPU platform has a physical limitation regarding the number of activities that can be simultaneously executed. For example, an NVIDIA GPU with a Pascal architecture and compute capability 6.1 can run simultaneous up to 32 activities, assuming there are enough resources to sustain their execution [10].

In this work, we assume that a GPU can execute simultaneous as many components as needed due to two main reasons: *i)* it is difficult to develop an embedded system that contains e.g., 32 independent GPU-aware components; and *ii)* even if the system has a high number of independent GPU-aware components, the GPU parallelism is limited by the component usage of hardware resources. For example, in a complex system that contains 10 independent GPU-aware components that have different image processing algorithms, only few of them can be executed in parallel due to the component high resource requirements, i.e., memory, threads and registers usage. The rest of the components wait for the resources to be released in order to be executed.

IV. System Description

The section describes our vision and assumptions on the software and hardware models of a CPU-GPU embedded systems.

A. The software system

The software application is composed of several components that communicate using various styles. For example, the Rubus component model follows the pipe-and-filter architectural style, where components are seen as filters that process data while the communication between components are pipes

that transfer data. Fig. 1 presents a general component-based system with GPU functionality, that follows the pipe-and-filter style and each of its GPU-aware component is characterized by various specifications. For example, component $C1$ is characterized by its quality (i.e., execution time) and EFPs as a set of GPU-specific requirements (e.g., usage of GPU memory). Because all GPU activities are triggered by the CPU, a GPU-aware component is characterized by both CPU and GPU specific properties. As our work targets the optimization of GPU activities execution, we focus only on the GPU-aware components and their GPU-specific properties, and discard the CPU-specific properties.

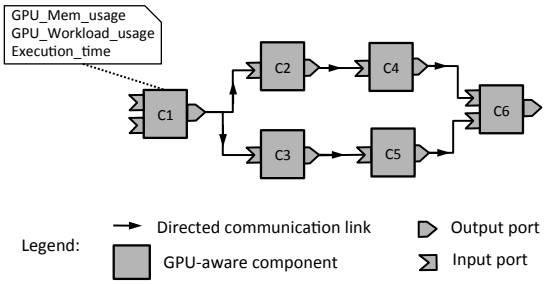


Fig. 1: System with GPU-aware components

In this initial work, we simplify and abstract the component GPU-specific properties as follows:

- the *GPU memory usage* specifies the component requirement of the GPU memory usage. The GPU has a hierarchical memory level, but we abstracted away several memory layers and use only the main memory level. The other memory layers are important factors and may be used in future work to extend our solution to a more detailed and precise software model.
- the *GPU workload* describes the GPU computation workload usage of the component; the metric utilized to describe this property is the number of computation threads. Several other factors related to the GPU workload are abstracted away (e.g., number of registers used by a thread).
- the *execution time* presents the time required by a component to fulfill its execution onto GPU.

B. The hardware system

Our focus being on GPU, we have abstracted away the CPU-specific properties (e.g., available RAM and CPU load) and characterized the hardware platform with only GPU-specific properties, as follows:

- the *available GPU memory* presents how much of GPU main memory a hardware is characterized by;
- the *GPU computation load* depicts the total of GPU computation threads a hardware is equipped with.

Similar with the software description, other influential GPU properties such as different memory levels (e.g., share memory) and the total amount of registers are removed in this introductory stage of our work.

V. Method Overview

In order to determine the components that may be executed in parallel and their execution order, the software and hardware properties are fed as input to our method. The hardware properties describe the platform resource limitation while the software specifications describe the component resource requirements and the communication pattern. The pattern of component communication has an important role when evaluating which component may run in parallel. For example, in Fig. 1, based on the communication pattern, we observe that $C2$ and $C4$ cannot be executed in parallel. $C4$ is depended of the $C2$ output and hence, they will be always sequentially executed.

The computed solution is expressed as a list composed of sublists of GPU-aware components. Each sublist contains components that can be, at a time instance, executed in parallel; the order of the sublists presents the order of the components execution.

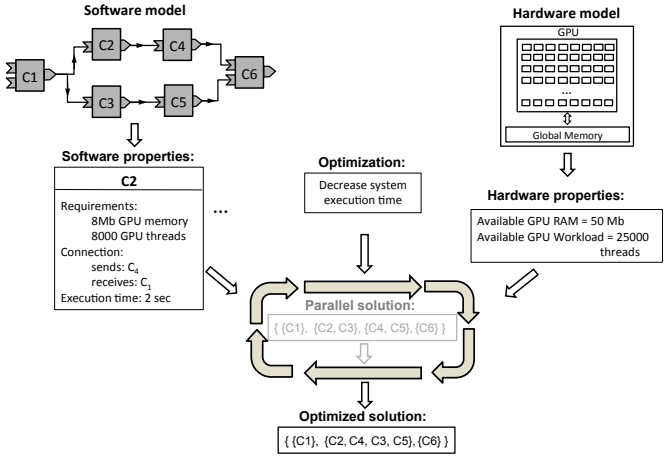


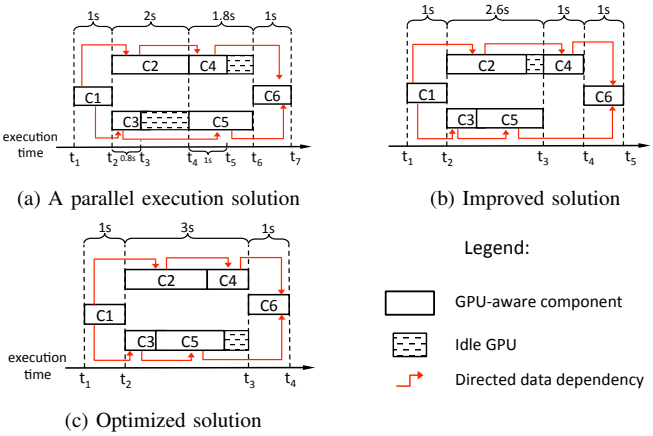
Fig. 2: The high-level overview of the proposed method

The solution overview is described in Fig. 2. Each component describes its resource requirements (e.g., GPU memory usage), how is connected to other components (i.e., to which component sends and from which component receives data) and its performance (i.e., execution time). The method calculates the components that can be executed in parallel, and their execution order.

Moreover, the method calculates an optimized parallel execution solution based on the component execution times. In order to describe the general idea of the optimization, we use the example with six connected components presented in Fig. 2. An initial solution is presented in Fig. 3(a) where, after $C1$ is executed, $C2$ and $C3$ are executed in parallel, followed by $C4$ and $C5$, and finally $C6$. The total execution time of the system is 5.8 seconds. Because $C5$ component is only dependent of $C3$ output data, and $C3$ has a shorter execution time than $C2$, an improved solution is to include $C5$ in the second batch [2] as it is seen in Fig. 3(b). This solution can be

[2]we will call the group of components that may be executed in parallel as a *batch* of components

further optimized when introducing $C4$ in the second batch, resulting an overall execution time of 5 seconds (Fig. 3(c)).



(a) A parallel execution solution

(b) Improved solution

(c) Optimized solution

Fig. 3: Optimization of a parallel execution solution

In general, we try to reduce the system overall execution time by possibly increasing and/or decreasing the execution time of different batches and possible reducing the number of batches. More precise, we look into batches where the GPU is idle and we try to fit in suitable components from the adjacent batch. For example, in Fig. 3(b) we migrated $C5$ in batch 2 because the GPU was idle (i.e., $C3$ has a short execution time) and $C5$ is only dependent of $C3$. We notice that, in the updated second batch (Fig. 3(c)), GPU is idle after $C5$ finishes, but we cannot fit in $C6$ because it is also dependent of $C4$ and there will be no improvement gain.

The optimization idea resembles with a bin packing problem, where we have already a number of bins and the solution is improved by re-fitting the bin sizes. This type of problem, where bin sizes are changed based on their items and their connection, is NP-hard [3], i.e., an exact solution cannot be calculated in feasible time (unless $P = NP$).

VI. Method Realization

This section presents the mathematical formalization of our method starting with the description of the (software and hardware) system followed by the system constraints and initial solution calculation, and finally the optimization step.

A. System definition

1) Let be $C = \{c_1, ..., c_n\}$ a set that contains a finite number of components with GPU functionality. Each component $c_i \in C$ is characterized by two multi-valued functions $Send : C \rightarrow C$, $Rec : C \rightarrow C$, and three single-valued functions $GPU\_Mem\_usage : C \rightarrow \mathbb{N}_{>0}$, $GPU\_Load\_usage : C \rightarrow \mathbb{N}_{>0}$ and $exec\_time : C \rightarrow \mathbb{Q}_{>0}$ described as follows:
   - *Send(c)* = a sublist that may contain none, one or several components that receive data from c;
   - *Rec(c)* = a sublist that may contain none, one or several components that send data to c;
   - *GPU_Mem_usage(c)* = the GPU memory usage of c;
   - *GPU_Load_usage(c)* = the number of GPU threads required by c; and
   - *exec_time(c)* = execution time of c.
2) The hardware is characterized by two constants: $GPU\_Mem \in \mathbb{N}_{>0}$ and $GPU\_Load \in \mathbb{N}_{>0}$, where:
   - *GPU_Mem* = the available GPU global memory;
   - *GPU_Load* = available number of GPU threads.

B. Constraints

- sum of the GPU memory usage of components from a batch (same sublist - see Section VI-C), cannot exceed the available GPU memory:

$$\sum_{c \in \{c | c \in C_i \subset C\}} GPU\_Mem\_Usage(c) \leq GPU\_Mem$$

- sum of the GPU thread usage of components from a batch, cannot exceed the available GPU available threads:

$$\sum_{c \in \{c | c \in C_i \subset C\}} GPU\_Load\_Usage(c) \leq GPU\_Load$$

C. Initial Solution Calculation

We see the solution as a list composed of sublists of components, i.e., $C = \{C_1, ..., C_k\}$. Each sublist contains components that can be parallelized; the sublist order presents the order of components execution. Each sublist is constructed based on its previous adjacent sublist as follows. We start by determining the first sublist that contains components with no input data; the following sublist contains components that receive data only from the components contained by the first sublist, and so on. In general, the solution list has two types of sublist elements:

- the first sublist type element $C_1 = \{c_p, ...\}$ that contains at least one element c_p and $Rec(c_p) = \varnothing$;
- the general sublist type element $C_k = \{c_q, ...\}$, where $\forall c_q \in C_k$, $Rec(c_q) \neq \varnothing$ and $Rec(c_q) \subset C_{k-1}$.

We mention that there exist only one first sublist type element while the general sublist type element may expand into none, one or several items. Some special cases may result in a solution with only one sublist item. For example, a system that contains one or several components that do not communicate among each other, will be executed in parallel (enclosed into one sublist element) if there are enough hardware resources.

D. Optimization

The initial solution calculated in the previous step is optimized (if possible) by decreasing the overall system execution. The overall idea, as described in Section V, is to look into batches where the GPU is idle and try to fit in components from an adjacent batch. Therefore, having a total of *k* batches $C = \{C_1, ..., C_k\}$, the sum of execution times of all batches is minimized:

$minimize\ ExTime = \sum_{i=1}^{k} cost_i$, where $cost_i$ represents the cost (i.e., the execution time) of a single batch C_i and is calculated by taking the largest cost (i.e., largest component execution time) from that batch:

$cost_i = \max_{\forall c_p \in C_i}(exec\_time(c_p))$.

Moreover, we may increase the cost of a batch by taking a component with (strictly) smaller cost than the batch's cost (i.e., a part of the GPU is idle) and adding to it the cost of a connected component from the adjacent batch:

$\forall c_j \in C_i,\ exec\_time(c_j) < cost_i,$
$exec\_time(c_j) = exec\_time(c_j) + exec\_time(c_q)$, where

c_q is a connected component $c_q \in C_{i+1}$,
$c_q \in Send(c_j)$ and all its connected components from batch i have a lower execution time than c_j:

$\forall c_m \in Rec(c_q) \wedge c_m \in C_i,$
$exec\_time(c_m) < exec\_time(c_j).$

The last condition is to ensure that the c_q component, when is added to the *i*-th batch, will not need to wait for the output data of another c_m component (from the same *i*-th batch) and will directly execute after c_j component finishes.

VII. Implementation

The optimization challenge is an NP-hard combinatorial problem [3]. Therefore, heuristic techniques need to be employed in order to find solutions. We selected the MINLP technique to address our method and to calculate feasible solutions. One solver that handles MINLP problems is SCIP [11], being one of the fastest non-commercial solvers existing on the market [12]. The solver divides the problem into smaller subproblems (know as *branching*) that are solved recursively. Moreover, for the solver to interpret our allocation model, we used the ZIMPL language [13] that translates the mathematical formulation into a readable format by SCIP. The following paragraphs briefly describe parts of our method implementation using the ZIMPL language.

Listing 1: Translation of the parallel execution model

```
1  set C := {"c1","c2","c3"};
2  param GPU_mem_use[C]:=<"c1"> 10,<"c2"> 80,<"c3"> 50;
3
4  set C_0 := { <c> in C where Rec(c) = 0 };
5  set C_1 := { <c> in C where Rec(c) in C_0 };
6
7  subto constraint: forall SubList in Sol  do
8   (sum <c> in SubList :GPU_mem_use[c]) <= GPU_mem_available;
9
10 minimize gpu_cost: sum SubList in Sol :
11  forall <c> in SubList do max(exec_time(c));
```

Besides the actual translation of the model constraints, it is required to construct a system model (in ZIMPL) in order to execute the model and find solutions. The system construction can be achieved in two ways: *i)* hard-coding the system and its characteristics directly into the ZIMPL program; and *ii)* reading the specifications from a file. The former mean is illustrated in Listing 1, where a system is defined as a set *C* of three components (line 1); the next line captures the GPU memory usage of each component. Similarly, we define the rest of the system specifications such as the available GPU memory and the component execution times.

We continue by constructing the initial solution that comprises of sublists of components. The first sublist *C_0* contains components that have no input data (see Section VI-C - first sublist type element); the following sublist, *C_1* is constructed based on the *C_0* content, i.e., all the components that receive data only from components of *C_0* sublist (lines 4 and 5). Regarding the implementation of constraints, we present the memory constraint, where the sum of the GPU memory usage of components from a sublist is less (or equal) than the available hardware memory (lines 7 and 8). The last part of the Listing describes a reduced form of the optimization function (due to the complexity structure), where we calculate the cost of each sublist *SubList* from the initial solution *Sol*, and minimize their summed cost (lines 10 and 11).

VIII. Running Case Evaluation

We examine our proposed method through a feasibility evaluation of an underwater robot demonstrator. The robot contains a CPU-GPU embedded board that communicates with actuators (e.g., thrusters) and sensors (e.g., cameras) [14]. Using the continuous feedback provided by cameras, the robot autonomously navigates underwater in fulfilling various missions (e.g., tracking red buoys). For designing the robot architecture, we used the Rubus component model [15] due to its fitting for developing streaming-of-event type of applications.

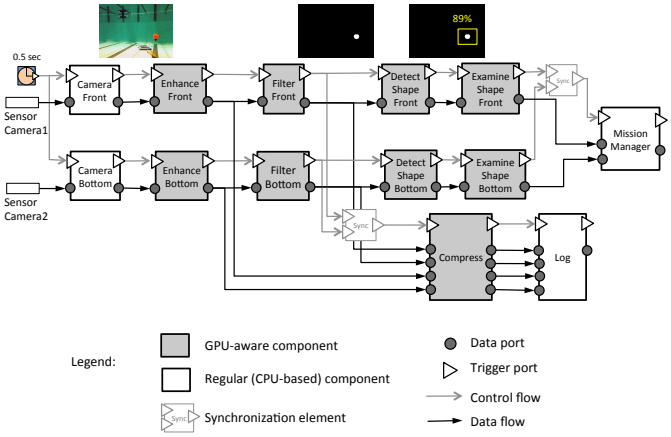


Fig. 4: The Rubus architecture of the vision system

Fig. 4 presents the vision system architecture of the robot. The physical (front and bottom) cameras provide raw data that is converted into readable frames by *CameraFront* and *CameraBottom*. These frames are forwarded to *EnhanceFront* and *EnhanceBottom* that improve the frames quality, by e.g., removing harsh edges. *FilterFront* and *FilterBottom* filter images using red-color criterion, resulting black-and-white frames. *DetectShapeFront* and *DetectShapeBottom* detect the shapes of the objects from the frames (e.g., circle, square) and forward their results to *ExamineShape* components that verify the found shapes against predefined shapes. The findings are sent to *MissionManager* that takes appropriate decisions. The *Compress* component compresses frames (e.g., resize) to allow

the (CPU) *Log* component to keep trace of the robot navigation for debugging purposes.

Each frame produced by cameras has a number of 600*400 pixels. When a frame is processed, we set that a GPU thread operates on 32 pixels. The GPU hardware has a total of 65536 threads and 128 Mb of memory. After translating the vision system using ZIMPL language and applying our proposed model on it, we obtain a solution illustrated by Table I.

TABLE I: The vision system execution scheme

Sublist (Batch)	GPU-aware component	Memory usage(Mb)	Thread usage	Execution time(ms)
1	Enhance Front	1.2	8000	35
	Enhance Bottom	1.2	8000	35
2	Filter Front	1.2	8000	30
	Filter Bottom	1.2	8000	30
3	DetectShapeFront	1.2	8000	45
	ExamineShapeFront	1.2	8000	45
	DetectShapeBottom	1.2	8000	40
	ExamineShapeBottom	1.2	8000	40
	Compress	4.8	24000	70

The solution orders the components into three batches of execution. We notice that the solution is optimized as batch 3 contains five components to be executed in parallel. An intermediate solution contains four batches, where in batch 3 only *DetectShapeFront*, *DetectShapeBottom* and *Compress* would be executed in parallel. Due to the high execution time of *Compress* and the hardware having enough resources, *ExamineShapeFront* and *ExamineShapeBottom* are migrated into batch 3.

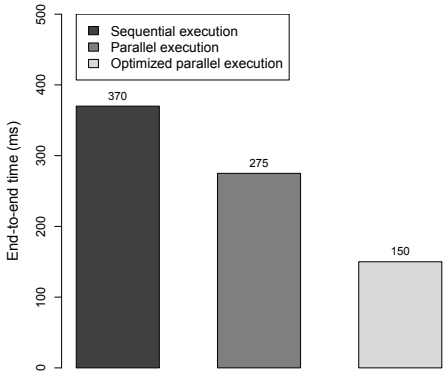


Fig. 5: Different execution times of vision system

Regarding the performance of the vision system, Fig. 5 presents the execution times when the system is executed in three cases. When the system is executed sequentially, the performance is the worst (i.e., 370 ms). An intermediate solution that executes the system in four batches, improves the system performance (i.e., 275 ms). Optimizing the solution and executing the system in three batches offers an enhanced performance (i.e., 150 ms).

By applying out method on the vision system case study, we improved the overall system performance. The specifications of our system allowed the method to provide an optimized solution.

IX. Related Work

Embedded systems embraced heterogeneity to improve system performance. Nowadays, we have platforms with multi-cores (e.g., SoC quad-core ARM processor [16]) and GPUs (e.g., AMD Carrizo APU). To manage the specifics of the new platforms, CBD introduced ways to handle the hardware particularities. We mention the work of Kopetz et al. [17] that explores the design alternatives of the AUTOSAR component model when targeting multi-core ECUs. The CPU-FPGA platforms are addressed by Andrews et al. [18] that introduce a way to utilize COTS components, by synchronizing CPU-FPGA computations inside the components. The Rubus component model is extended with new artifacts (e.g., GPU ports), to allow efficiently development of CPU-GPU embedded systems [19].

Due to an increased complexity and challenging quality requirements, system optimization approaches has proliferated. There is a body-of-knowledge presented in different surveys [20] [21], that targets optimization of software architectures from different system domains (e.g., embedded systems and information systems) considering various quality attributes. Yet, there is a reduce amount of work that addresses the optimization of component-based architectures of embedded systems with different computation nodes. We mention the work of Campeanu et al. that targets component-based systems with many CPU and GPU computation nodes [22]. The authors allocate components over hardware considering their (CPU and GPU) requirements and optimize the allocation based on different criteria. The drawback of this work is that it does not take in attention the architecture design and considers that all GPU-aware components can be parallelized at once. In our work, we consider how the components are connected and optimize their parallel execution on a single GPU node.

Specific scheduler solutions for GPU-based systems are addressed by various works. We mention Muyan-Özçelik et al. [23] that developed several scheduling algorithms for (GPU) tasks, where a task is defined as a series of operations, i.e., a host-to-device copy, a GPU (kernel) execution and a device-to-host copy. For example, an algorithm increases the system performance by scheduling copy operations in the same time as GPU (kernel) executions. The work presents the findings of its schedulers using NVIDIA technology. By targeting only NVIDIA GPUs, some of their claims are based on specific hardware technology which may not hold for GPUs from other vendors. Moreover, they have a high control over scheduled tasks. In our work, one or several tasks may be incorporated into a component and we do not have the same level of control over them. For example, we do not know when a device-to-host copy operation (encapsulated into a component) ended in order to give the control to a GPU (kernel) execution (encapsulated by another component).

Moreover, even if we create such mechanisms, the context-switching between components would be very expensive. Basically, it would worth to have these mechanisms when components would have very large copy operations and GPU (kernel) executions. Other works use the same (refined) control level in order to provide parallelization strategies. We mention a two-level parallelization strategy that works directly with the GPU (kernel) functionality by e.g., analyzing its loop iterations and their statements [24].

As our optimization challenge is similar to the bin-packing problem, we want to mention the surgical scheduling problem, where the operating rooms seen as bins, can change their (time) capacity by increasing the number of operations [25]. In addition, we find the multiprocessor scheduling problem [26] related to our challenge and we refer specifically to the Gang scheduling [27] which is considered to be an efficient algorithm for parallel and distributed systems. One of its types is Bags of Gangs (or Bags of Tasks) [28] in which the jobs, considered as independent gangs that belong to a bag, are sent to be executed by the system. A bag finishes its execution only when all of its gangs finish.

X. Conclusions

This work introduces an initial method that addresses the parallel execution of components on GPU. Our proposed method computes execution schemes by considering: *i)* hardware characteristics (e.g., available GPU memory); *ii)* software constraints (e.g., required number of GPU threads); and *iii)* component communication pattern. The method optimizes the computed schemes w.r.t. performance (i.e., execution time) resulting in schemes with maximum degree of component parallelism. Being an NP-hard combinatorial problem, the optimized schemes are calculated by using a MINLP heuristic method. The last part of our work presents the feasibility aspect of the proposed method when is applied on an underwater robot case study.

To the best of our knowledge, there are no developed component mechanisms to execute GPU-aware components in parallel. In addition, as several components may simultaneously access the GPU, mechanisms to protect the GPU (seen as a shared resource in this context) need also to be developed. We consider these aspects as future directions of our work.

Acknowledgments

The Swedish Foundation for Strategic Research (SSF) supports our work inside the RALF3 project (IIS11-0060).

References

[1] I. Crnkovic and M. Larsson, *Building Reliable Component-Based Software Systems*, 2002.

[2] T. Henzinger and J. Sifakis, "The Embedded Systems Design Challenge," in *Proceedings of the 14th International Symposium on Formal Methods*, 2006.

[3] S. Anily, J. Bramel, and D. Simchi-Levi, "Worst-case analysis of heuristics for the bin packing problem with general cost structures," *Operations research*, 1994.

[4] S. A. Manavski, "CUDA compatible GPU as an efficient hardware accelerator for AES cryptography," in *IEEE International Conference on Signal Processing and Communications. ICSPC 2007.*

[5] J. E. Stone, J. C. Phillips, P. L. Freddolino, D. J. Hardy, and L. G. Trabuco, "Accelerating molecular modeling applications with graphics processors," *Journal of computational chemistry*, 2007.

[6] S. N. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Genc, "GPU-based video feature tracking and matching," in *EDGE, Workshop on Edge Computing Using New Commodity Architectures*, 2006.

[7] M. Humenberger, C. Zinner, M. Weber, W. Kubinger, and M. Vincze, "A fast stereo matching algorithm suitable for embedded real-time systems," *Computer Vision and Image Understanding*, 2010.

[8] Arcticus Systems, "Customers," http://www.arcticus-systems.com/links/, accessed: 2017-03-01.

[9] I. Crnkovic, S. Sentilles, A. Vulgarakis, and M. R. Chaudron, "A classification framework for software component models," *IEEE Transactions on Software Engineering*, 2011.

[10] "NVIDIA CUDA C programming guide," https://docs.nvidia.com/cuda/cuda-c-programming-guide/, accessed: 2017-03-01.

[11] G. Gamrath, T. Fischer, T. Gally, A. M. Gleixner, G. Hendel, T. Koch, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, S. Schenker, R. Schwarz, F. Serrano, Y. Shinano, S. Vigerske, D. Weninger, M. Winkler, J. T. Witt, and J. Witzig, "The SCIP optimization suite 3.2," ZIB, Tech. Rep., 2016.

[12] H. Mittelmann, "Mixed integer linear programming benchmark," http://plato.asu.edu/ftp/milpc.html, accessed: 2017-03-01.

[13] T. Koch, "Rapid mathematical prototyping," Ph.D. dissertation, Technische Universitat Berlin, 2004.

[14] C. Ahlberg, L. Asplund, G. Campeanu, F. Ciccozzi, F. Ekstrand, M. Ekstrom, J. Feljan, A. Gustavsson, S. Sentilles, I. Svogor *et al.*, "The Black Pearl: An autonomous underwater vehicle," 2013.

[15] Arcticus Systems, "Rubus models, methods and tools," http://www.arcticus-systems.com, accessed: 2017-03-01.

[16] ARM, "The ARM Cortex-A53 processor," https://www.arm.com/products/processors/cortex-a/, accessed: 2017-03-01.

[17] H. Kopetz, R. Obermaisser, C. El Salloum, and B. Huber, "Automotive software development for a multi-core system-on-a-chip," in *Workshop of Software Engineering for Automotive Systems*, 2007.

[18] D. Andrews, D. Niehaus, and P. Ashenden, "Programming models for hybrid CPU/FPGA chips," *Computer*, 2004.

[19] G. Campeanu, J. Carlson, S. Sentilles, and S. Mubeen, "Extending the Rubus component model with GPU-aware components," in *19th Int. Symposium on Component Based Software Engineering*, 2016.

[20] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya, "Software architecture optimization methods: A systematic literature review," *IEEE Transactions on Software Engineering*, 2013.

[21] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni, "Model-based performance prediction in software development: A survey," *IEEE Transactions on Software Engineering*, 2004.

[22] G. Campeanu, J. Carlson, and S. Sentilles, "Component allocation optimization for heterogeneous CPU-GPU embedded systems," in *The 40th Euromicro Conf. on Soft. Eng. and Advanced Applications*, 2014.

[23] P. Muyan-Özçelik and J. D. Owens, "Multitasking real-time embedded GPU computing tasks," in *Proceedings of the 7th International Workshop on Programming Models and Applications for Multicores and Manycores*. ACM, 2016, pp. 78–87.

[24] J. Shirako, A. Hayashi, and V. Sarkar, "Optimized two-level parallelization for GPU accelerators using the polyhedral model," in *Proceedings of the 26th International Conference on Compiler Construction*. ACM, 2017, pp. 22–33.

[25] J. H. May, W. E. Spangler, D. P. Strum, and L. G. Vargas, "The surgical scheduling problem: Current research and future opportunities," *Production and Operations Management*, 2011.

[26] M. R. Gary and D. S. Johnson, "Computers and intractability: A guide to the theory of NP-completeness," 1979.

[27] J. K. Ousterhout, "Scheduling techniques for concurrent systems," in *ICDCS*, 1982.

[28] W. Cirne, F. Brasileiro, J. Sauve, N. Andrade, D. Paranhos, E. Santos-neto, and R. Medeiros, "Grid computing for bag of tasks applications," in *In Proc. of the 3rd IFIP Conference on E-Commerce, E-Business and EGovernment*, 2003.

New Optimal Solutions for Real-Time Scheduling of Operating System Tasks Based on Neural Networks

Ghofrane Rehaiem
Faculty of Mathematical, Physical and Natural Sciences of Tunis (FST), Tunis El Manar University, TUNISIA
LISI-INSAT Laboratory (INSAT), Carthage University, TUNISIA
E-mail: rh.ghofrane@gmail.com

Hamza Gharsellaoui
National Engineering School of Carthage (ENIC), Carthage University, Tunisia
LISI-INSAT Laboratory (INSAT), Carthage University, TUNISIA
E-mail: gharsellaoui.hamza@gmail.com

Samir Ben Ahmed
Faculty of Mathematical, Physical and Natural Sciences of Tunis (FST), Tunis El Manar University, TUNISIA
LISI-INSAT Laboratory (INSAT), Carthage University, TUNISIA
E-mail: samir.benahmed@fst.rnu.tn

[1]

***Abstract*—This paper work deals with the implementation of a neural networks based approach for real-time scheduling of embedded systems composed by Operating Systems (OS) tasks in order to handle real-time constraints in execution scenarios. In our approach, many techniques have been proposed for both the planning of tasks and reducing energy consumption. In fact, a combination of Dynamic Voltage Scaling (DVS) and time feedback can be used to scale the frequency dynamically adjusting the operating voltage. In this study, Artificial Neural Networks (ANNs) were used for modeling the parameters that allow the real-time scheduling of embedded systems under resources constraints designed for real-time applications running. Indeed, we present in this paper a new hybrid contribution that handles the real-time scheduling of embedded systems, low power consumption depending on the combination of DVS and Neural Feedback Scheduling (NFS) with the energy Priority Earlier Deadline First (PEDF) algorithm. Experimental results illustrate the efficiency of our original proposed approach.**

***Index Terms*—Optimization, Neural Networks, Real-Time Scheduling, Low-Power Consumption.**

I. Introduction

The main use of neural networks is to classify possible results, from a list of informations. More generally, a neural network allows the approximation of a function. Let's take the example of a processor, it can adjust its speed in a continuous range to ensure that a task is submitted to, at most, a fault occurrence, and the cost of restoration of state is zero. In this paper, we want to discuss these issues to propose a heuristic approach based on neural networks (NN) which can handle real-time constraints allowing the real-time scheduling with minimization of power consumption of real-time embedded systems. A neuron is then the elementary processing unit of a neural network. It is connected to sources of information as input and returns output information. All these are organized through the learning method. In new technology, the use of fixed parameter method has allowed the development of most embedded detective devices, what is known as the mapping relationship between test input data and test conclusion. In this work, we must develop and implement an optimal approach that considers the following issues in the design which are the choice of the:

- network model,
- the number of input and output nodes,
- the number of hidden layer nodes,
- the choice of transfer function,
- the choice of initial weight,
- the choice of the learning rules,
- the preprocessed of input and output data in training sample in order to have a good solution balancing between the real-time scheduling and the low power consumption optimization of the embedded systems at the same time.

In this paper, Section 2 gives an overview of the neural network platform design and the neural feedback scheduling. Also, we describe the real-time dynamic voltage loop scheduling DVS. In Section 3, we present our PEDF heuristic and in Section 4 we shows our experimentation results. Finally, Section 5 concludes the paper work.

II. Background and Related Works

Neural network which can adapt the sample data by training has good fault-tolerance and can be used in the field of intelligence widely. In the embedded systems, restricted to the resources and the capacity of processor, the neural network application has some problems such as losing timeliness, also the system could be collapsed easily.

A. Neural Network Platform Design:

In practical applications, the input conditions of detection and the testing standards may need to be modified, which can damage the existing detection system. The neural network owns the ability of self-adaptation and self-learning. Therefore, by studying on the sample data, it can determine the mapping

[1]DOI reference number: 10.18293/SEKE2017-025

relationship between the input and the output and if the neural network is applied to the detection system, it may adjust the mapping relationship automatically by training samples, which can make the detection system more flexible and adaptable [(Bernard, 2008)]. The following paragraph is about the description of issues needed to be considered in the design.

1) The Choice of Network Model: The 2-layer linear perceptron model and the 3-layer Back Propagation (BP) network model are provided and the user can choose one of them in the network parameter settings in accordance with the complexity of the detection system. Based on the [(Hagan, 1996)] works, a 3-layer BP neural network with a hidden layer can approximate to any continuous function of bounded domain with arbitrary precision as long as there are enough hidden layer nodes. Therefore, in terms of the function, 3-layer BP neural network can meet most sophisticated detection systems' requirements, while for some detection systems which are simple mapping, linear perceptron model with a small amount of calculation and fast speed can be chosen so as to maximize the efficiency of the systems.

2) The Choice of the Number of Input and Output Nodes: The number of the input nodes is determined by testing item. The number of the output nodes is generally determined by the number of the testing conclusions. But, if there are a lot of testing conclusions, the number of the output nodes could become great, then, the amount of calculation also increases. So, when there are a lot of testing conclusions, the testing conclusions can be encoded to binary code, then output nodes take the number greater than or equal to $log_2(N)$, where N is the number of conclusions [(Liu, 2011)].

3) The Choice of the Number of Hidden Layer Nodes:: If the network model is Back Propagation (BP) network, the number of the hidden layer nodes needs to be set. Generally speaking, if there are too few hidden layer nodes, the network can not study well, and much more training will be needed and, meantime the training will not be highly precise; while too many nodes will bring issues such as a large amount of calculation, long training time and reduced network fault-tolerance capacity. System uses the default settings as the empirical formula.

$$n_1 = \sqrt{n+m} + 2 \tag{1}$$

In the formula 1, n_1 is the number of the hidden layer nodes; n is the number of the input nodes; *m* is the number of the output nodes. If the training effect of experience value is not good, we can reset the number of the hidden layer nodes in the parameter settings and do many experiments in order to achieve faster convergence rate. In order to know the training efficiency, you should output the training number in training process, changes of the network errors, changes of network weights and the training time, so as to make a report for the network analysis [(Liu, 2011)].

4) The Choice of Transfer Function: In the application of transfer function detection, most of the outputs are the field data and test conclusions. The field data are collected by the acquisition system; detection conclusions are obtained by the aplication of the neural network algorithm. The detection conclusions are indicated with two-value, so the transfer function of output layer generally uses the sigmoid function [(Liu, 2011)] which is presented as follows:

$$f(x) = \frac{1}{1+e^{-x}} \tag{2}$$

The transfer function is a function that should return a real close to 1 in the presence of "good" input information and real close to 0 when they are "bad". Generally, we use functions with values in the interval of real [0, 1]. When the real is close to 1, the neuron is called "active" whereas when the real is close to 0, we say that the neuron is inactive. The real in question is called the output of the neuron. If the activation function is linear, the neural network would be reduced to a simple linear function. The sigmoid function has the advantage of being differentiable as well as giving intermediate values (real between 0 and 1). However, this function have a threshold that is 1/2 when x = 0.

5) The Choice of the Learning Rules: Due to the particularity of embedded systems, we should try to select the learning algorithm with simple calculation and low memory consumption. For this reason, and to improve the efficiency, a flexible BP learning algorithm may be adopted. It was only needed to consider the symbol of gradient to the error function, rather than the increasing amplitude of the gradient. The symbol of gradient determines the direction of weight updating, and the change of the weight size is determined by an independent "update value" [(Hagan, 1996)]. The iterative process of weight correction example is shown below:

$$\omega(t+1) = \omega(t) + \alpha\Delta\omega(t) \times sgn\frac{\partial E_t}{\partial \omega_t} \tag{3}$$

In the formula 3 [(Tian, 2009)], $\Delta\omega(t)$ is the previous update value, and the initial value $\Delta\omega(0)$ is set by the actual application. α is the learning parameter, using variable step-size learning.

6) The Preprocessing of Input and Output Data in Training Sample: In the network learning process, because the transfer function of the neuron is a bounded function, while in the detection, as the dimension and unit of input test item data may be different, some values are very large, while others are very small, which has a great influence on the network. To prevent some neurons from reaching saturation state, and meantime make the larger input fall in the region where gradient of neuron activation function is large, input and output vectors need to be normalized before the training.

$$x_i' = \frac{x_i - x_{imin}}{x_{imax} - x_{imin}} \tag{4}$$

In the formula 4, x_{imax} and x_{imin} are the maximum value and the minimum value of each input component of the neuron number *i*; x_i, x_i' are respectively the former and later input component after pretreatment of the neuron number *i*. The input data values after normalization processing range between

0 and 1. If the neural network training is unsuccessful, or if the training time is too long, process of training must be analyzed to train after readjusting the network parameters. If the training results are satisfactory, the testing sample data may be used to test the function of the network. Successful testing demonstrates that the construction of the network is completed, and then the network parameters and the weights of each layer of the network are saved as files, ready for testing procedures.

B. Neural Feedback Scheduling

The basic idea of the neural feedback scheduling (NFS) is to allocate available resources dynamically among multiple real-time tasks based on feedback information about actual resource usage. To tackle the problem associated with the large computational overheads of optimal feedback scheduling algorithms, a NFS scheme must be proposed. The goal is to optimize the overall Quality of Control (QoC) of multitasking control systems through feedback scheduling while minimizing the feedback scheduling overhead [(Xia, 2008)].
On one hand, this scheme has advantages such as low feedback scheduling overhead, wide applicability, and intelligent computation. It is also capable of delivering almost optimal QoC. On the other hand, neural networks are powerful in learning and adapting, and capable of approximating complex nonlinear functions with arbitrary precision [(Hagan, 1996)], [(Zhu, 2006)]. Once well trained using the accurate optimal solutions at design time, neural networks will be able to deliver online almost-optimal feedback scheduling performance.

C. Real-time Dynamic Voltage Loop Scheduling

Low power is extremely important for real-time embedded systems. A real-time loop scheduling techniques were proposed to minimize energy consumption via Dynamic Voltage Scaling (DVS) for applications with loops considering transition overhead. Two algorithms, Iterative Dynamic Voltage Scheduling (IDVS) and Dynamic Voltage Loop Scheduling (DVLS), are designed integrating with dynamic voltage scaling (DVS). IDVS is an algorithm to iteratively optimize the Directed Acyclic Graph (DAG) part of a loop by incorporating transition overhead into optimization scheduling scheme. DVLS is an algorithm to repeatedly regroup a loop based on rotation scheduling and decreases the energy by DVS as much as possible within a timing constraint. DVS is one of the most powerful techniques to reduce energy consumption by adjusting supply voltage at running time.

III. CONTRIBUTION

In this section, we describe our proposed approach and we give its notation to be more clear. So, to do this, we will present how to use processor and a limited memory to perform a neural network algorithm in order to improve adaptability in real-time embedded systems. In the application of the neural network, we need to enter a large number of training data and output of the training process and the results of the training, which requires a higher demand for input and output. Our main goal is about minimizing the number of neurons in order to facilitate the implementation which will be adapted later. In this context, we must ensure a low complexity and fast convergence and as a consequence the number of neurons must be significantly reduced. Therefore, new building regulations have been proposed to design the smallest possible neural network in order to optimize the scheduling of reconfigurable embedded systems in real-time

A. Formalization

Embedded applications are implemented in a complex SOC (System-on-Chip). Other resources, such as cores or dynamically reconfigurable accelerators need to be controlled by the OS. In particular, an instantiation of the task execution resources is performed using the OS scheduling service. As each task can be defined for multiple targets, this service has to decide at run-time, on what resource the task must be instantiated. Neural networks have demonstrated their efficiency in optimization constraint problems with the ability to converge in a reasonable time (i.e., few cycles) if the number of neurons and connections between them can be limited as much as possible. It should be noted that when the network converges to a stable state which does not belong to the set of valid solutions, this network need to be reinitialized.

1) Scheduling Modeling Through Neural Network:: In context of SoC architecture, service implementations for task scheduling are often complex and are not always suitable for real-time systems because they are usually time costly and they do not consider the dynamic behavior of the application. Our solution uses Artificial Neural Networks (ANN) for online real-time scheduling, where we have chosen the Hopfield model [(Hopfied, 1985)] to ensure network convergence to a stable state, while respecting the optimization constraint. This function is defined as follows:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{i,j} \times x_i \times x_j - \sum_{i=1}^{N} I_i \times x_i \qquad (5)$$

- $T_{i,j}$ is the connection weight between the neurons *i* and *j*.
- x_i is the state of the neuron *i*.
- I_i is the external input of the neuron *i*.

Based on this model and by using an optimization function of the constraints, a design rule can be defined in order to facilitate construction of the neural network. The rule k-out-of-N is a major result in ANN for optimization. This rule allows the construction of *N* neurons for which the evolution leads to a stable state with "exactly *k* active neurons among *N*". The energy function is defined as follows:

$$E = (k - \sum_{i=1}^{N} x_i)^2 \qquad (6)$$

This function is minimal when the active neuron sum is equal to k, and is positive otherwise. The results of this scheduling solution in real-time demonstrate the interesting

convergence speed which makes ANN suitable for online utilization. However, this technique has two major weaknesses. The first is the large number of slack neurons needed to model the problem, which depends on the cycle, so that when the schedule time increases, the number of slack neurons also increases. The second problem is the presence of several local minima when many rules are applied to the same set of neurons. These local minima are particular points of the energy function representing invalid solutions. Our work is to considerably reduce the number of hollow neurons for any period. The principal consequence is the simplification of network control. It is clear at this point that our proposition is guided by a main objective which consists of reducing the number of neurons while ensuring the convergence of the network.

∗ **Monoprocessor architecture:**

In the case of monoprocessor architecture, the scheduling problem is modeled through ANN by the following representation:

Neurons n_{ij} are organized in a matrix form, with the size N_T x N_C, where line *i* represents the task τ_i and the column *j* corresponds to schedule time unit *j*. The number of time units N_C is the least common multiple of all the task periods and N_T is the number of tasks.

- A neuron n_{ij} is considered active when the task τ_i is being executed, during the corresponding time unit *j*.
- One line of neurons is added to model the possible inactivity of the processor during the schedule times. These neurons are called slack neurons.

∗ **Multiprocessor architecture:**

In the case of homogeneous multiprocessor architecture, several matrices arranged in layers are required to model the different execution resources. New slack neurons are then needed to manage the execution of each task on resources. For each couple (task τ_i, resource *j*), $C_{i,j}$ new slack neurons should be added. So, the total number of slack neurons is equal to:

$$\sum_{i=1}^{N_T} \sum_{j=1}^{p} C_{i,j} + p \times N_C \tag{7}$$

Running example of network with *p* resource layers is presented in Figure 1. Grey circles represent slack neurons.

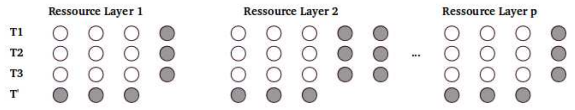


Fig. 1. Classical Structure to Model the Scheduling Problem with ANN.

To reduce the required number of neurons, we must modify neural network structure. The changes correspond to an adaptation of the Hopfield model. We have adopted the idea of creating a mutual exclusion between the possible task instantiations on execution resources [(Chillet, 2007)]. This mutual exclusion is provided by the presence of an inhibitory neuron $IN_{i,j}$ for the task τ_i and the execution of resource *j*. In Figure 2, an example of the scheduling problem is presented with one task τ_i and *R* possible resources.

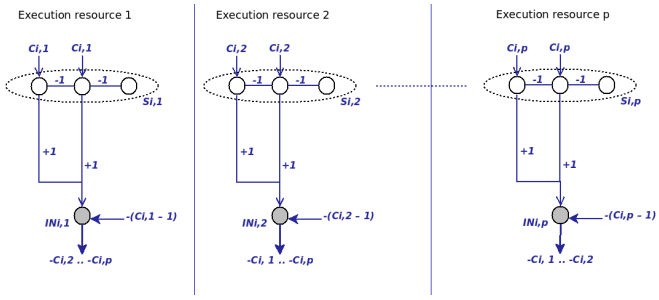


Fig. 2. Scheduling Problem Modeled with Inhibitor Neurons.

A set of N_C neurons is called $S_{i,j}$ (N_C = 3 in our use case example) and represents the possible scheduling cycles of the task τ_i on the resource *j*. For each resource j, the Worst Case Execution Time (WCET) of the task τ_i is defined as $C_{i,j}$. The set of neurons S_j is configured (definition of inputs and weights) to converge towards $C_{i,j}$ active neurons among N_C. The main characteristic of this neuron network is its capacity to converge to a stable state from any initial state. One or more lines of slack neurons can be added to ensure the network convergence during the application of k-out-of-N rule on each vertical line of neurons. As shown in Figure 1, the number of added lines in each layer is equal to the number of identical processors in this layer (In our example, one resource can execute the tasks, so one line of slack neurons are added, line T'). In this case, the convergence can not be always obtained.

To delete these lines, we choose the application of a k1-out-of-N1 classical rule on the horizontal sets of neurons and a at-most-k2-among-N2 rule on the vertical sets of neurons. If N_T tasks must be scheduled on *p* identical resources (*p* processors in the same layer) during the N_C cycles, N_C at-most-*p*-among-N_T rules must be applied on each vertical set of N_T neurons.

There remains the problem concerning the application of two rules on both sets of neurons with a common neuron. Figure 3 shows an example of this case, where the first set of neurons is composed of three horizontal neurons n_1, n_2, n_3, and the second is composed of three vertical neurons n_1, n_4, n_5 (Neuron n_1 is the common neuron of the two sets). The classical additive for various rules mentioned that if a $k1$-out-of-3 rule is applied on the first set and at-most-$k2$-among-of-3 rule is applied on the second set, then the inputs and weights are defined as follows:

-Inputs are equal to:

$I_1 = (2 \times k_1 - 1) + (2 \times k_2 - 1)$

$I_2 = I_3 = (2 \times k_1 - 1)$

$I_4 = I_5 = (2 \times k_2 - 1)$

- Weights are equal to $w_{i,j} = -2$; $\forall i, j = 1..5$

$k2$ slack neurons can be added with a specific weight

connection with other neurons in order to ensure the at-most-$k2$-among-of-3 rule. In the figure 3, the slack neurons (S_{n1}, S_{n2}) are represented.
To remove the slack neurons while ensuring convergence, we need to simplify the k-out-of-N rule by adopting the simple redefinition of input and weight values. The energy function given in the equation 6, energy was rewrited as $E = (\frac{1}{\sqrt{2}}k - \frac{1}{\sqrt{2}}\sum_{i=1}^{N} x_i)^2$.
With these new input values and weight, we can suggest a simple additive between $k1$-out-of-$N1$ and at-most-$k2$-among-$N2$ rules. The main idea is to apply the rule $k1$-out-of-$N1$ rule at first on horizontal lines and $k2$-out-of-$N2$ rule on the vertical lines and secondly the change of weight of horizontal lines.
Thus, the common neuron has its input set at the value $k1 + k2$ as shown in Figure 3. The change of the weight values of the horizontal rule (here the horizontal ruler $k1$-out-of-3) compensates the increase of input values on the common neuron. This compensation is done by decreasing the weight value between the horizontal neurons and the common neuron between the two rules. An example of additive of the two rules is given in Figure 3. We present

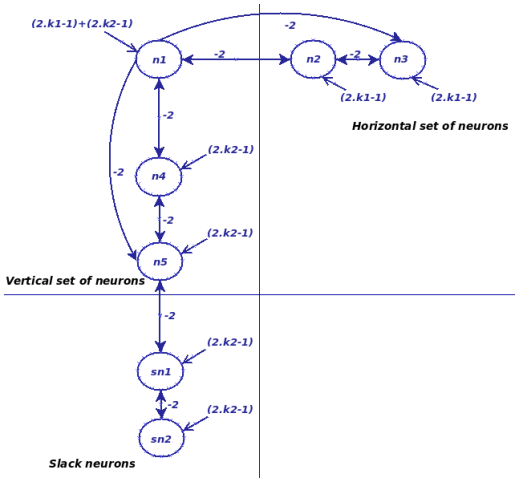


Fig. 3. Additive of k_1-out-of-N_1, and at-most-k_2-among-N_2 Rules with Slack Neurons.

an online scheduling algorithm for real-time systems that attempts to minimize the energy consumed by a periodic task set. This is based on the well-known earliest-deadline-first (EDF) algorithm [(Liu, 1973)]. We attempt to find the voltage at which each task must be executed such that the energy consumed by the entire set of periodic tasks is minimized and generate a schedule for the task set such that the release time requirements are satisfied and the deadline for each task is met. Although the scheduling methods cited above are very efficient, most of them make the assumption that the Central Processor Unit (CPU) can operate at several different voltage levels (and hence different clock frequencies) which can be varied continuously. In addition, a number of these methods are aimed at the synthesis of low-power designs and they do not address energy minimization during field operation.

The optimization problem that we address in this work, is to minimize the total energy consumed by the set of *n* tasks by optimally determining their start times, their voltages and corresponding execution speeds at the real-time scheduling. The following constraints need to be modeled:
(i) CPU speeds are limited to one of three values S_1, S_2 and speed-medium $S_3 = S_m = (S_1 + S_2)/2$,
(ii) The deadline of each task must be met,
(iii) Tasks are non preemptable,
(iv) A task may start only after it has been released.
The objective function is to minimize $\sum l_i v_i^2$. The modeling of the constraints and their subsequent linearization are omitted here due to space limitations.

2) The PEDF Heuristic:: The Priority-energy Earliest Deadline First heuristic, or simply PEDF, is an extension of the well-known earliest deadline first (EDF) algorithm. The operation of PEDF is as follows: PEDF maintains a list of all released tasks, called the ready list. When tasks are released, the task with the nearest deadline is chosen to be executed. A check is performed to see if the task deadline can be met by executing it at the lower voltage (speed). If the deadline can be met, PEDF assigns the lower voltage and the execution of the task begins. During the task's execution, other tasks may arrive at any random time. These tasks are assumed to be placed automatically on the ready list. PEDF again selects the task with the nearest deadline to be executed. As long as there are tasks waiting to be executed, PEDF does not keep the processor idle. This process is repeated until all the tasks have been scheduled.

Begin algorithm
Procedure PEDF
Repeat forever
If tasks waiting to be scheduled are in ready list
Sort deadlines in ascending order
Schedule task with earliest deadline
If deadline can be met at lower speed (voltage)
schedule task to execute at lower voltage (speed)
If deadline can not be met at medium speed (voltage)
schedule task to execute at medium voltage (speed)
If deadline can not be met at higher speed (voltage)
schedule task to execute at higher voltage (speed)
Else deadline cannot be met, task cannot be scheduled.
End algorithm

3) Notation:: We present the notation and the underlying assumptions. Let *T* be a set of assumed *n* periodic tasks where *T* = τ_1, τ_2,...,τ_n. Associated with each task $\tau_i \in T$ are:
(i) its arrival time a_i,
(ii) its deadline d_i,
(iii) its length l_i (represented in number of instruction cycles),
(iv) its period p_i.
Each task is released at time t = a_i. Release times are arbitrary where each task may be released at any time before its deadline. We assume that the Central Processor Unit (CPU) can operate at one of the three voltages: V_1, V_2 or V_3.

Depending on the voltage level, the CPU speed may take on three values: S_1, S_2 or S_3. The supply voltage to the CPU is under OS control and the OS may dynamically switch the voltage during run-time with relatively low overhead. CPU speeds are specified in terms of the number of instructions executed per second. Each task τ_i may be executed at a voltage V_i, $Vi \in \{V1, V2, V3\}$.

Task	Arrival Time a_i	Deadline d_i	Length l_i	l_i/v_1	l_i/v_2	l_i/v_3
τ_1	3	7	800	2.66	2	2.28
τ_2	9	21	750	2.5	1.875	2.14
τ_3	0	5	1600	5.33	4	4.57
τ_4	18	25	1000	3.33	2.5	2.85
τ_5	14	16	600	2	1.5	1.71
τ_6	7	10	1200	4	3	3.42
τ_7	20	27	1100	3.66	2.75	3.14
τ_8	14	20	1600	5.33	4	4.57
τ_9	11	14	500	1.66	1.25	1.42

TABLE I
TASK SET COMPOSED OF 9 TASKS.

IV. EXPERIMENTATION RESULTS

We present now our experimental results. First, we show the results of the PEDF for a task set of nine tasks. Our example task set is given in Table I. It consists of tasks τ_1 to τ_9. Each task has a release time a_i, a deadline l_i and a length l_i. We assume that the three processor speeds are 300 million instructions per second (MIPS) at 2.47 V, 350 MIPS at 2.885 V and 400 MIPS at 3.3 V. The energy consumed by the schedule generated through MILP is 75677,4575 units (measured by the sum of the $l_i v_i^2$ values).

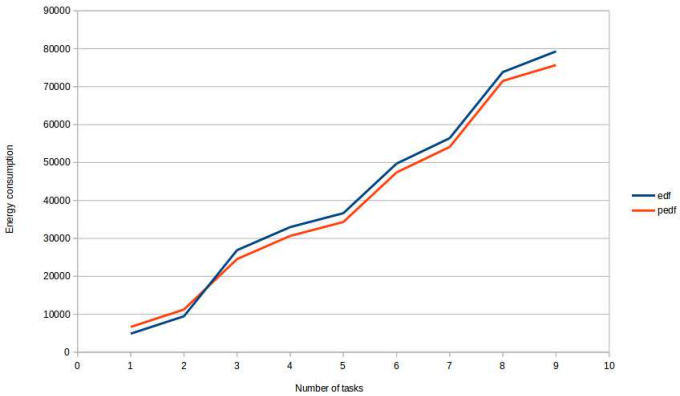


Fig. 4. Comparison of Schedules Generated by EDF and PEDF

In this part, we observe that the increased energy consumption of EDF arises due to the fact that EDF does not possess knowledge of the release times a priori. Our PEDF model, which does not have such a restriction, executes tasks τ_3 and τ_9 at a medium speed (voltage) even though both could have met their respective deadlines by executing at a lower speed (voltage). We can observe that energy consumed by a task is proportional to its length. Since the length of task τ_3 is greater than the lengths of both τ_1 and τ_2. We observe that by executing τ_1 at a medium speed (voltage), and τ_2 at a higher speed (voltage), we can execute τ_3 at a medium speed (voltage) and thus reduces the effect of the length of τ_3 on energy consumption. This, in fact, does result in an optimal schedule. The results, plotted in Figure 4, prove that PEDF produces optimal schedules.

Now we use the first row from the above I, (3, 7, 800) to demonstrate forward propagation: For this example, we take two hidden layers with six neurons. Then, we assign weights to all of the synapses. These weights are selected randomly (based on Gaussian distribution). The initial weights will be between 0 and 1.

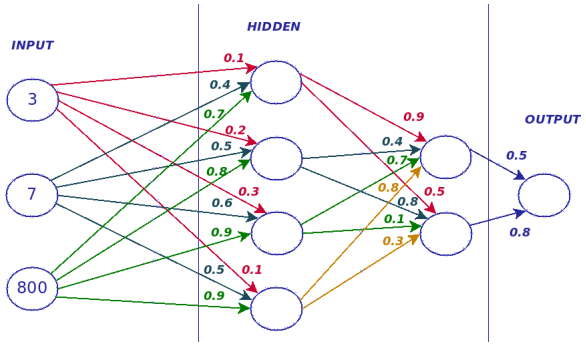


Fig. 5. Determination of Initial Weights

We calculate the sum of the product of the inputs with their corresponding set of weights to arrive at the first values for the hidden layer. Weights may be considered as measures of influence of input nodes on the output.

3 * 0.1 + 7 * 0.4 + 800 * 0.7 = 563.1
3 * 0.2 + 7 * 0.5 + 800 * 0.8 = 644.1
3 * 0.3 + 7 * 0.6 + 800 * 0.9 = 724.2
3 * 0.1 + 7 * 0.5 + 800 * 0.9 = 723.8

We put these sums smaller in the circle, because they are not the final values:

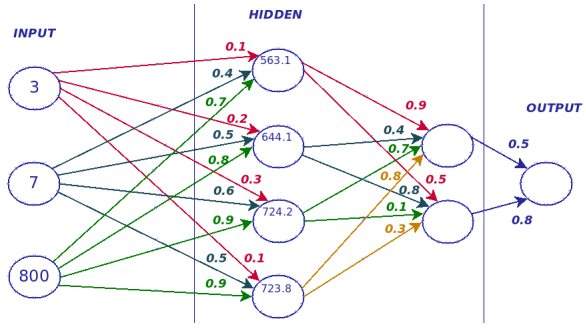


Fig. 6. Calcul of Sums for the First Hidden Layer

We apply the activation function 2 to the hidden layer sums. The purpose of the activation function is to transform the input signal into an output signal and are necessary for neural networks to model complex non-linear patterns. We use the sigmoid function as Transfer Function $f(x) = \frac{1}{1+e^{-x}}$:

f(563.1) = f(644.1) = f(724.2) = f(723.8) = 1
We add the obtained results to our neural network as hidden layer results: Then, we sum the product of the hidden layer

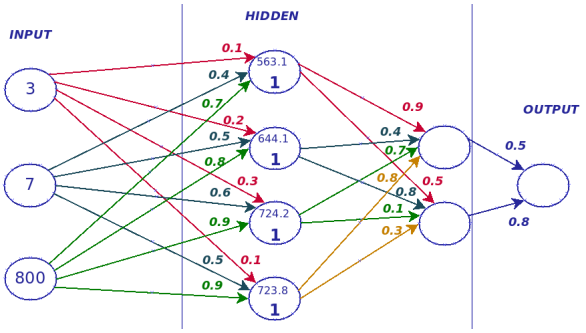


Fig. 7. Application of activation function

results with the second set of weights (also determined randomly the first time around) to determine the output sum.
1 * 0.9 + 1 * 0.4 + 1 * 0.7 + 1 * 0.8 = 2.8
1 * 0.5 + 1 * 0.8 + 1 * 0.1 + 1 * 0.3 = 1.7
We apply now the Transfer Function:
f(2.8) = 0.94
f(1.7) = 0.84

Following the presented method, we had the final output result.

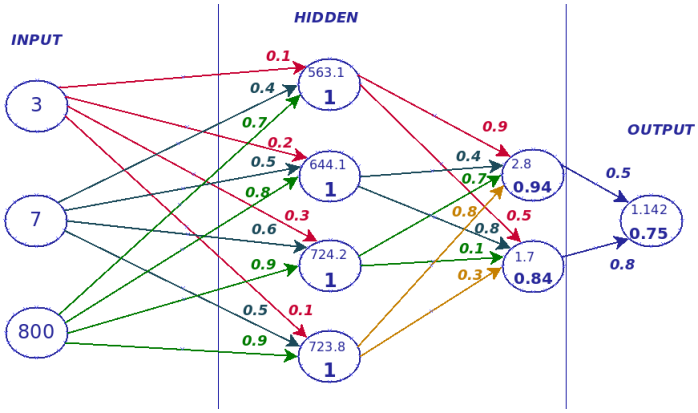


Fig. 8. Full Diagram

V. Conclusion

In this paper we have presented advanced approaches for real-time scheduling of reconfigurable embedded systems to meet real-time constraints in different execution scenarios using neural networks. As a fast and intelligent feedback scheduling scheme, neural feedback scheduling has been proposed in this paper for real-time scheduling of OS tasks. To minimize the total energy consumed, we integrated the PEDF as online scheduling algorithm for real-time systems.
A first perspective for the future works is to extend our original approach based on a neural network approach and especially on the back propagation method to support real-time embedded systems failures.

References

[(Bernard, 2008)] W. Bernard, D.S. Samuel: ”Adaptive Signal Processing”. Machinery Industry Press, Beijing, 2008.

[(Liu, 2011)] D. Li, Y. Liu, and Y. Chen. ”The Application Research of Neural Network in Embedded Intelligent Detection”. (Eds.): CCTA 2010, Part IV, IFIP AICT 347, pp. 376381, 2011.

[(Tian, 2009)] Y. Tian. ”Hybrid Neural Network Technology”. Science Press, Beijing, 2009.

[(Hagan, 1996)] M.T. Hagan, H.B. Demuth and M.H. Beale. ”Neural Network Design”, PWS Publishing, USA, 1996.

[(Xia, 2005)] F. Xia, S.B. Li and Y.X. Sun. ”Neural Network Based Feedback Scheduler for Networked Control System with Flexible Workload”, Int. Conf. on Natural Computation (ICNC), Lecture Notes in Computer Science, vol.3611, pp.237-246, 2005.

[(Weiser, 1994)] M. Weiser, B. Welch, A. Demers and S. Shenker. ”Scheduling for reduced CPU energy”, Proc. Sympo-sium on Operating System Design and Implementation, pp. 1323, 1994.

[(Liu, 1973)] C. L. Liu and J. Layland. ”Scheduling algorithms for multiprogramming in a hard real-time environment”, Journal of the ACM, vol. 20, pp. 4661, 1973.

[(Xia, 2008)] F. Xia, Feng and Tian, Yu-Chu and Sun, Youxian and Dong, Jinxiang, Neural feedback scheduling of real-time control tasks. International Journal of Innovative Computing, Information and Control (IJICIC), 4(11). pp. 2965-2975, 2008.

[(Zhu, 2006)] E. Zhu, Z.B., A simple feasible SQP algorithm for inequality constrained optimization, Applied Mathematics and Computation, vol.182, pp.987-998, 2006.

[(Hopfied, 1985)] J. J. Hopfield and D. W. Tank. Neural computation of decisions in optimization problems. Biological Cybernetics, 52:141-52, 1985.

[(Chillet, 2007)] Daniel Chillet, Sebastien Pillement and Olivier Sentieys. A Neural Network Model for Real-Time Scheduling Heterogeneous SoC Architectures on Heterogeneous SoC Architectures.Proceedings of International Joint Conference on Neural Networks, Orlando, Florida, USA, August 12-17, 2007.

Model Construction and Data Management of Running Log in Supporting SaaS Software Performance Analysis

Rui Wang[1,2], Shi Ying[1,2*], Chengai Sun[3], Hongyan Wan[1,2], Huolin Zhang[1,2], Xiangyang Jia[1,2]
1. State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China
2. School of Computer, Wuhan University, Wuhan 430072, China
3. College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China
{wangrui1989,yingshi}@whu.edu.cn, sun910213@163.com, {why0511,huolinzhang,jxy}@whu.edu.cn

***Abstract*—Changes in operating environment may result in the performance degradation to a SaaS software. Analyzing running log is an efficient method to locate this problem. However, as a long-running software, SaaS may generate huge log data which is difficult to analyze, and it lacks a systematic approach to implement management of the running log. These all threaten the timeliness of SaaS performance analysis. In this paper, we define a log format to standardize multi-source heterogeneous log data and construct a log model to support SaaS software performance analysis, where the two performance metrics of average response time and request timeout rate in the model are calculated by statistical measurement. Furthermore, a log management framework is given to support real-time big log data collection, access, calculation, storage and service, and the technology implementation of the framework is also given. Finally, a case study is given to illustrate and validate the effectiveness of the approach.**

Keywords-big data; running log; log model; SaaS software performance

I. INTRODUCTION

SaaS software provides software as service to consumers, while service quality is the key factor for consumer satisfaction. Performance as an important QoS attribute of SaaS software, directly affects the user experience. In the dynamic, scalable running environment provided by cloud computing, if the average time that SaaS software responds to service request, especially from tenant is too long, when the service does not meet the service level agreement, and even losses availability, we believe that it is suffering a performance issue (PI). When this happens, it tends to cause consumer dissatisfaction, even cause the loss of consumers, and finally damage the interests of service providers.

After SaaS software is deployed and put into operation, it will provide the highest possible service throughput with the lowest possible response latency in various running environments and usage scenarios. Due to the development quality of the software system, the complex changes of the running environment and its resources, the diversity of usage scenarios and loads, etc., it will cause the software to suffer a variety of performance issues in the runtime. In addition, high performance is the basic requirement of SaaS software. With the increasingly large number of users, business process execution time will get longer and longer, and the possibility of SaaS software performance issues will also grow.

Therefore, at the runtime of SaaS software, operation and maintenance managers need find performance issues timely, exhaustively, accurately, and take measures to ensure that the system can restore the invalid service to the available state in time, make the software continue to provide high performance services. In this case, through corresponding methods and facilities getting a general idea of the performance state of a system becomes a necessary condition to achieve above demand.

The running log of the SaaS software is the data that records the information about the status, events, processes or changes in the software and its operating environment, the usage behavior of users, the occurred events, the interactive messages, and other aspects. Running log is widely applied in the various tasks managed by software system, such as software failure analysis, running environment analysis, usage behavior analysis, etc. The operational information extracted from the running log can help operation and maintenance managers to analyze software status, there are two main means: One is to detect the anomaly of the software by analyzing the log features, for example, Wei Xu[1] proposed an automatic method that extracts information from console logs to detect and visualize runtime problems in large scale distributed systems. Deqing Zou et al.[2] proposed a method that classifies log according to different failure types, which can assists system administrators to monitor the operation of the system and diagnose the failure. Fu X et al.[3] presented a methodology and a system, named LogMaster, for mining correlations of events that have multiple attributions based on system logs and predicting system failure. The other is to analyze the performance of the software by analyzing the performance counters, for example, Keith A. Bare et al.[4] proposed an online diagnostic framework ASDF that transparently monitors and analyzes different time-varying data sources (e.g., OS performance counters, Hadoop logs), and narrows down performance issues to a specific node or a set of nodes. Zhiling Lan et al.[5] proposed a set of techniques to automatically analyze collected data from per node, to detect the nodes acting differently from others in the large scale distributed system, in another reference[6], they adopted a divide-and-conquer approach to address the scalability challenge emerging in this problem and explored the use of non-parametric clustering and two-phase majority voting to improve detection flexibility and accuracy.

However, the above studies mainly focus on the analysis of the characteristics of the log itself to detect software anomalies or monitoring the data in the PaaS or IaaS layer to analyze the

*Corresponding Author
DOI reference number: 10.18293/SEKE2017-128

software performance, few studies have been done on the performance monitoring and analysis of Web application components even fine-grained services in the SaaS layer. The features of SaaS software in the cloud computing environment, lead to large and complex log data related to performance which have high requirement for analysis and processing. So it is difficult to identify and diagnose the performance issues of SaaS software and has high timeliness requirements. Although the application level of big data technology continues to improve, the research on the analysis and processing technology of big log data is not yet mature, and the research on the analysis of SaaS software performance issues based on running log is especially lacking. Therefore, how to scientifically and effectively analyze and utilize big log data, correctly obtain performance-related log data, further diagnose occurred performance issues, and how to use big data technology efficiently record, process and analyze big log data become a challenging problem in log data analysis and processing.

For these above problems, in view of the SaaS software and its cloud computing environment, combined with the demand for SaaS software performance analysis, this paper studies how to construct log model and manage the log data supporting SaaS software performance analysis. We first define the log format to normalize the log distributed on each virtual machine node, then construct log model and calculate the performance metrics in the model via statistical measurement method, and finally design and implement the whole framework for managing raw log data and log model data.

This paper is organized as follows. In Section 1, we introduce the research background of this paper. In Section 2, we introduce the method of constructing log model. In Section 3, we explain our log management framework and discuss the technology implementation of the framework. Our case study is presented in Sections 4. We conclude our work in Section 5.

II. LOG MODEL CONSTRUCTION BASED ON STATISTICAL MEASUREMENT

A. Performance Metrics

Performance is generally used to define and measure the time constraints and resource allocation degree of a software system efficiency. Common performance metrics include response time, throughput, resource utilization, etc., the first two reflect the performance status of the application level, the latter reflects the performance status of the server level.

In this paper, from the application level, the two metrics of response time and throughput are obtained through collecting and constructing of the performance metric data from SaaS software running log. Performance monitoring of SaaS applications belongs to high-level monitoring[7]. In order to describe the performance of SaaS software at high level, the performance metrics are transformed into the values of statistical measurement. In this paper, according to the definition of two traditional software performance metrics, response time and throughput, we obtain the two performance metrics supporting SaaS software performance analysis: 1) Average Response Time (ART), that reflects the time efficiency that service processes requests. The longer the ART, the slower service processing requests, the lower the performance; 2) Request Timeout Ratio (RTR), which reflects the degree of service overload. The higher the RTR, the higher load, the lower the performance.

B. Log Format

Context information at SaaS software runtime is generally obtained by the way of logging. We define logging rules as follows: 1) Service Invocation Begin (SBE), the SBE event has to be logged as the first instruction (begin) of the service. The event, if logged, provides the trace that the entity initiated the invocation of the service; 2) Service Invocation End (SEN), the SEN event notifies the termination (end) of the service and has to be logged immediately before each normal exit point of the service. The event, once logged, provides the trace that the entity completed the invocation of the service; 3) Service Timeout Error (STE), the STE entry is written in the event log when a service timeout is detected and reported to system.

Log components generally employ binary instrumentation[8] to log, but the log records generated by this method has the characteristic of discontinuity. We solve this problem by adding a unique identifier GUID in the log records. We use the JSON format as the log format standard. An example of log format is shown in Fig. 1.

```
{
 "WebAppName":"D2Mgr",
 "VHostIP":"172.17.15.165",
 "GUID":"81225c51-bcad-4a4a-b467-e0f0acddc602",
 "Timestamp":"2017-02-08 10:05:18,814",
 "Level":"INFO",
 "ServiceName":"ImageRegisterService",
 "Content":"SBE"
}
```

Figure 1. An example of log format.

And we adopt the time window method to analyze the SaaS software running log[9]. The size of the time window can be set according to actual demand.

So GUID and TimeStamp are the core fields of the log model supporting SaaS software performance analysis. The work of this paper focuses on analyzing the performance of SaaS software, we need monitor the performance of each service component even fine-grained service method in cloud environment, so the SaaS software running log should also include the location information field about SaaS service component even service method: Web component name and IP address of virtual machine. The basic log information field should include service name (service method name), log level (log type) and log context (context execution information).

C. Log Model

The raw log data generated by SaaS software at runtime has the characteristics of volume, velocity, value and veracity (4V). For big data processing technology, there are many applications in other fields[10-11]. While the log is a stream of data that is a real-time, continuous, time-varying sequence of data items. Data stream model is the logical abstract expression of the data stream, which can improve the processing efficiency of data streams. In order to improve the efficiency of big log data processing and analysis, we propose a log model, it mainly consists of 7 attributes, as shown in Fig. 2.

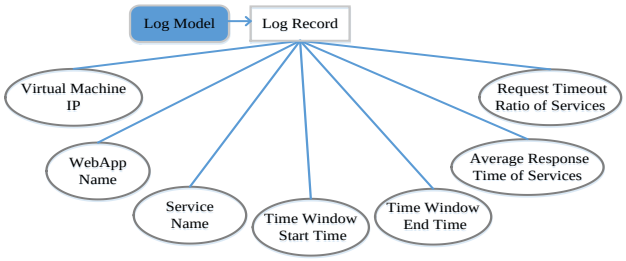


Figure 2. Log model supporting SaaS software performance analysis.

We calculate the two performance metrics of Average Response Time and Request Timeout Ratio in the model by statistical measurement method.

1) Average Response Time Calculation

Response time is the end-to-end time that a task spends to traverse a certain path within the system[12]. Average Response Time reflects the user expectation on time that software responds to request. The ART is computed as

$$ART_w = E[T_{rw}] = \sum_{i=1}^{n}(t_{ri} - t_{si})/n \quad (1)$$

where W is the time window, it can be a given time interval, such as 1 minutes, 1 hours, 1 days etc.; T_{rw} is the response time of request r in W; n is the number of requests in W; t_{ri} indicates the arrival time of the request i, that is the time of SBE; t_{si} indicates the service response time of the request i, that is the time of SEN. This calculation does not take into account the time consumption about network transmission from client to server. Equation (1) is applied only in the case that each (SBE, SEN) pair in a time window.

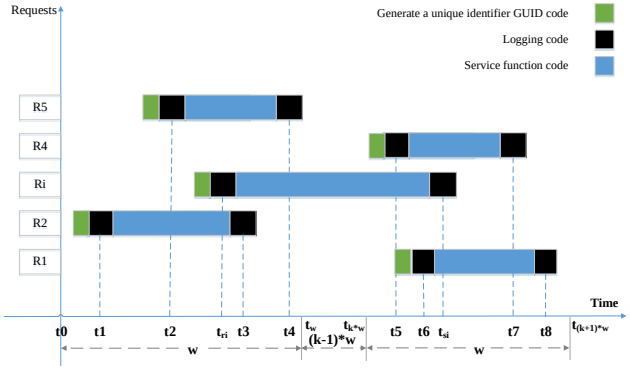


Figure 3. The requests of across the time windows.

For example in Fig. 3, request Ri across multiple time windows from request arrival to service response. We consider the proportion of time spent on each time window. We divide the Ri into three parts, calculate the ART for each part. The ART of request Ri in the time window $[t_0, t_w]$ is $\frac{t_w - t_{ri}}{(t_w - t_{ri})/(t_{si} - t_{ri})}$, in the time window $[t_{kw}, t_{(k+1)w}]$ is $\frac{t_{si} - t_{kw}}{(t_{si} - t_{kw})/(t_{si} - t_{ri})}$, in the time interval $[t_w, t_{kw}]$ is $\frac{(k-1)w}{(k-1)w/(t_{si} - t_{ri})}$.Therefore, the ART for all requests in the time window$[t_0, t_w]$ is $\frac{(t_3 - t_1) + (t_4 - t_2) + (t_w - t_{ri})}{1+1+(t_w - t_{ri})/(t_{si} - t_{ri})}$, in the time window $[t_{kw}, t_{(k+1)w}]$ is $\frac{(t_7 - t_5) + (t_8 - t_6) + (t_{si} - t_{kw})}{1+1+(t_{si} - t_{kw})/(t_{si} - t_{ri})}$.

Combined with the above example, we modify (1) as

$$ART_{w_j} = \frac{\sum_{i=1}^{n_j} T_{r_i W_j}}{\sum_{i=1}^{n_j} K_{r_i W_j}} \quad (2)$$

where W_j is a time window from t_{jw} to $t_{(j+1)w}$, i.e. $W_j = [t_{jw}, t_{(j+1)w}]$; n_j is the number of requests within W_j; $T_{r_i W_j}$ is the time proportion of ith request r_i in W_j, i.e.

$$T_{r_i W_j} = \begin{cases} t_{si} - t_{ri}, & t_{jw} < t_{ri} < t_{si} < t_{(j+1)w} \\ t_{si} - t_{jw}, & t_{ri} < t_{jw} < t_{si} < t_{(j+1)w}; \\ t_{(j+1)w} - t_{ri}, & t_{jw} < t_{ri} < t_{(j+1)w} < t_{si} \end{cases}$$

$K_{r_i W_j}$ is the number of ith request r_i in W_j, i.e.

$$K_{r_i W_j} = \begin{cases} 1, & t_{jw} < t_{ri} < t_{si} < t_{(j+1)w} \\ \frac{t_{si} - t_{jw}}{t_{si} - t_{ri}}, & t_{ri} < t_{jw} < t_{si} < t_{(j+1)w} \\ \frac{t_{(j+1)w} - t_{ri}}{t_{si} - t_{ri}}, & t_{jw} < t_{ri} < t_{(j+1)w} < t_{si} \end{cases}.$$

2) Request Timeout Ratio Calculation

Request Timeout Ratio describes that the service accepts multiple requests simultaneously, but responses to some requests exceed the user expectation in a given time window, then the percentage of these requests is the request timeout ratio.

In time window W_j, the RTR is the ratio of the number of timeout requests N_{tW_j} to the total number of requests N_{aW_j}:

$$RTR_{W_j}(\Delta t) = \frac{N_{tW_j}}{N_{aW_j}} \quad (3)$$

Because it is a ratio, the metric is less influenced by isolated extreme values, which makes it more independent of time window length. But N_{tW_j} and N_{aW_j} can not be directly obtained through the program, we modify (3) as

$$RTR_{W_j}(\Delta t) = \frac{N_{tW_j}}{N_{aW_j}} = \frac{\sum_{i=1}^{n_j} N_{tW_j}^{r_i}}{\sum_{i=1}^{n_j} N_{aW_j}^{r_i}}, \sum_{i=1}^{n_j} N_{aW_j}^{r_i} \neq 0 \quad (4)$$

where $N_{tW_j}^{r_i}$ indicates whether ith request r_i timeouts within W_j,

$$N_{tW_j}^{r_i} = \begin{cases} 1 \; t_{jw} < t_{ri} < t_{si} < t_{(j+1)w}, t_{si} - t_{ri} > \Delta t \\ 0 \qquad\qquad otherwise \end{cases},$$

1 indicates that the request timeouts within the window.

$N_{aW_j}^{r_i}$ indicates whether ith request r_i is within W_j,

$$N_{aW_j}^{r_i} = \begin{cases} 1 \; t_{jw} < t_{ri} < t_{si} < t_{(j+1)w} \\ 0 \qquad\qquad otherwise \end{cases},$$

1 indicates that the request is within the window.

A request timeout belongs to the STE event, we estimate the expected duration Δt of the requested service timeout by EWMA[13]. The timestamps of the SBE and SEN events, logged at each exception-free invocation, are used to profile Δt. The EWMA statistic as described in the following:

$$\Delta_n = (1 - \alpha)\Delta_{n-1} + \alpha\Delta_l \quad (5)$$

where Δ_l is the last duration estimate (obtained by subtracting the timestamp of the SBE from the one of the SEN event at the last service invocation); α is the weight, the weight is small for taking into account the history in the changing tendency of the Δ_n series. A requested service timeout is detected if the SEN event is not observed within $\Delta t = n_s \cdot \Delta_n$ time units since the SBE. Then the STE event will be written in the log and reported to system.

The structure of log model is fixed as shown in Fig. 2, but log model data is calculated in real time with the generation of raw log data, it has the characteristic of rapid growth. In addition, the log model is for different roles, and the requirements for the log model are different due to different demands of the roles. In Fig. 2, we define the core attributes of a log model that supports the diagnosis of performance issue that response time is too long. In the actual diagnosis, we can add the specific attribute items to meet the different requirements of specific diagnostic tasks of different roles.

III. LOG MANAGEMENT SUPPORTING SAAS SOFTWARE PERFORMANCE ANALYSIS

A. Log Management Framework

The log data management framework adopts the idea of modularization[14].

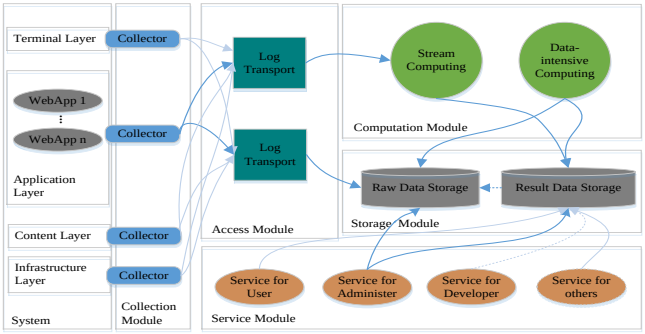


Figure 4. The log management framework supporting SaaS software performance analysis.

Fig. 4 shows the log management framework that supports the processing of big log event stream data in real-time. It consists of five modules: collection module, access module, computation module, storage module and service module, the functions of each module are described as follows:

1) Log Collection Module

This module is responsible for collecting data in real-time from virtual machine nodes on the cloud platform. As the log data is multi-sourced and heterogeneous, each collector logs the data generated by different objects and normalizes the collected log data according to the log format defined in this paper;

2) Log Access Module

This module is responsible for easing the situation that the speed of data collection and data processing is not synchronized, and solving the cost problem that immediately process collected data. So it is necessary to transfer the log data to the storage module or computation module for centralized processing;

3) Log Computation Module

In this module, one is data-intensive computing applied to analyze massive raw log data to obtain the required information. The other is stream computing applied to process the coming data in real-time. Furthermore, this module is extensive for its computing frameworks based on actual analysis demands;

4) Log Storage Module

This module includes two kinds of storage systems. One is raw data storage, which stores historical data orderly and permanently for future data mining and analyzing. The other is result data storage, which provides rapid access to data for different role analysis, evaluation or prediction;

5) Log Service Module

This module consists of services that can meet the different data demands of different roles. Each service reads required data from the Result Data Storage for all roles in each layer. In addition, to meet the demand of an administrator for diagnosing performance issues, the raw log data in Raw Data Storage are needed.

B. Implementation of Log Management Framework

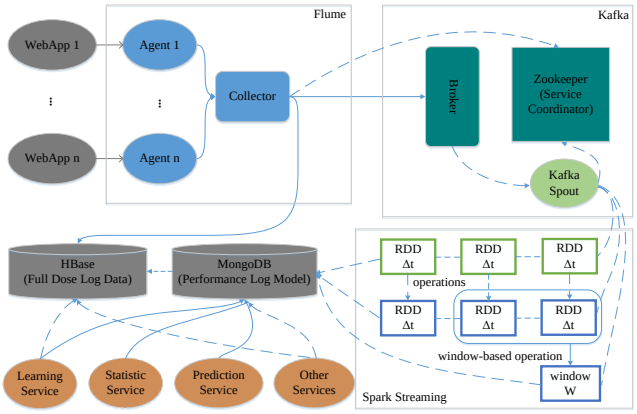


Figure 5. The log management framework implementation supporting SaaS software performance analysis.

As shown in Fig. 5, event log generated from the SaaS application WebApp deployed on the virtual machine in cloud platform will be collected by Agent of the Flume collection process of each virtual machine, then it flows into the Flume sink node of a virtual machine and generates two log event stream branch: The first flows directly to the HBase database for storage; The second will flow into the Spark Streaming for calculation, and the results are written into the MongoDB database.

1) Log Collection Module Implementation

We collect the log context of SaaS software at runtime by interceptor technology, as shown in Fig. 6.

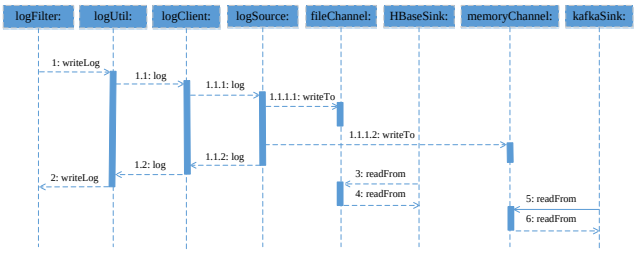


Figure 6. Log collection sequence diagram.

2) Log Access Module Implementation

We adopt Kafka mainstream framework as distributed message queue middleware, as shown in Fig. 7.

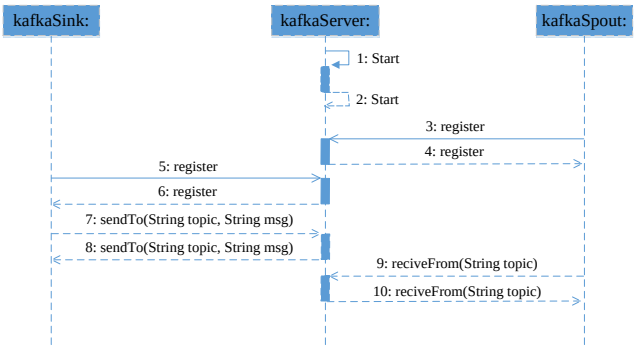


Figure 7. Kafka's producers and consumers publish subscribe message sequence diagram.

3) Log Computation Module Implementation

We implement the real-time calculation of log data stream based on Spark Streaming[15], as shown in Fig. 8.

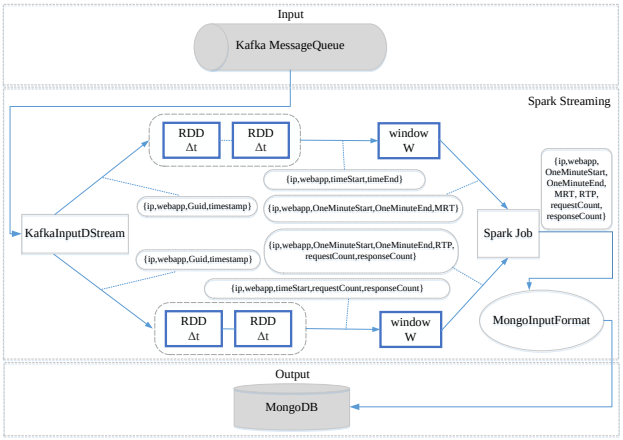


Figure 8. Log data stream real-time computing process.

Spark Streaming provides some basic log data statistics interface, we use its basic statistical functions: total, mean and percentage. For the window operation of Spark Streaming, we use incremental methods to improve the efficiency of statistics.

4) Log Storage Module Implementation

a) log data storage

Raw log data generally has no high-frequency access and has no high requirement for real-time, so we consider storing it on Hadoop Distributed File System (HDFS). This paper stores the raw log data according to the log level in the log record;

b) log model data storage

Log model data will be accessed frequently by the log service module. Combined with the characteristics of the log model itself, this paper chooses to store it in the extensible, high-performance distributed NoSQL database MongoDB. In order to improve the efficiency of query processing, we establish B-tree index[16] at the SBE time for log model data with the increasing computational requirements.

5) Log Service Module Implementation

The module provides the basis for operation and maintenance managers to find software performance issues, and improve the issues effectively. For the implementation of this module, from discovering performance issues to timely improving performance issues, demand of the roles in SaaS software may involve three services: historical log data statistics service, real-time monitor service and prediction service.

IV. CASE STUDY

We evaluate the effectiveness of our method by an Integrated Disaster Reduction Application System (IDRAS). It relies on the basic idea of cloud service platform and service oriented architecture (SOA), has a series of Web components with independent functions, as shown in Fig. 9.

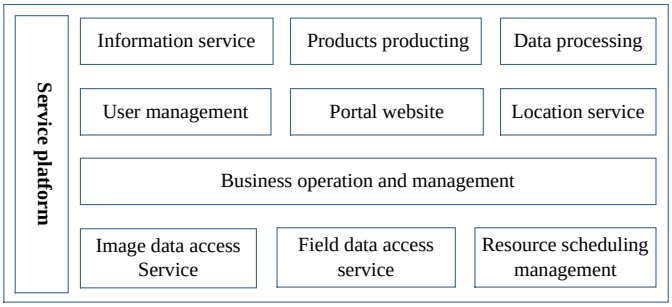


Figure 9. The service platform of IDRAS.

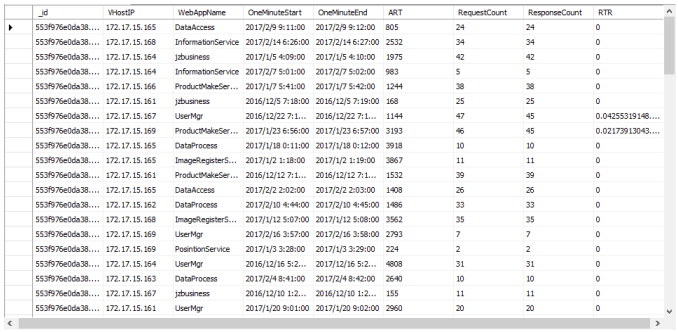
When the disaster occurs, all the SaaS software services in the system work together to provide services. In such a complex production environment, SaaS application will produce hundreds of MBs of log data with millions of events in them. Full manual analysis of these log data is not a realistic option, yet they need to be used daily by the operation and maintenance managers to monitor and resolve performance issues. Therefore, it is of great significance to provide a unified log management system for the massive running log generated by IDRAS to support its performance analysis. And our log management system can handle these hundreds of MBs of log data even larger.

We create a collection called LogModel in the MongoDB, the document structures of the collection include Key, Value, and Type. Table I elaborates the types and descriptions of each key in the log model in the document structure.

TABLE I. LOG MODEL DOCUMENT STRUCTURE DESCRIPTION

Key	Type	Description
_id	ObjectId	Primary key field
VHostIP	String	Mobile IP or fixed IP
WebAppName	String	such as war files
OneMinuteStart	DateTime	Start time
OneMinuteEnd	DateTime	End time
ART	Int32	Average Response Time
RequestCount	Int32	The number of requests in 1min
ResponseCount	Int32	The number of responses in 1min
RTR	Double	1-(ResponseCount/RequestCount)

We visualize the log model data results supporting IDRAS performance analysis through MongoVUE, Fig. 10 shows a small part of the data.



_id	VHostIP	WebAppName	OneMinuteStart	OneMinuteEnd	ART	RequestCount	ResponseCount	RTR
553f976e0da38....	172.17.15.165	DataAccess	2017/2/9 9:11:00	2017/2/9 9:12:00	805	24	24	0
553f976e0da38....	172.17.15.168	InformationService	2017/2/14 6:26:00	2017/2/14 6:27:00	2532	34	34	0
553f976e0da38....	172.17.15.164	jzbusiness	2017/1/5 4:09:00	2017/1/5 4:10:00	1975	42	42	0
553f976e0da38....	172.17.15.164	InformationService	2017/2/7 5:01:00	2017/2/7 5:02:00	983	5	5	0
553f976e0da38....	172.17.15.166	ProductMakeSer...	2017/1/7 5:41:00	2017/1/7 5:42:00	1244	38	38	0
553f976e0da38....	172.17.15.161	jzbusiness	2016/12/5 7:18:00	2016/12/5 7:19:00	168	25	25	0
553f976e0da38....	172.17.15.167	UserMgr	2016/12/22 7:1...	2016/12/22 7:1...	1144	47	45	0.04255319148....
553f976e0da38....	172.17.15.169	ProductMakeSer...	2017/1/23 6:56:00	2017/1/23 6:57:00	3193	46	45	0.02173913043....
553f976e0da38....	172.17.15.165	DataProcess	2017/1/18 0:11:00	2017/1/18 0:12:00	3918	10	10	0
553f976e0da38....	172.17.15.165	ImageRegisterS...	2017/1/2 1:18:00	2017/1/2 1:19:00	3867	11	11	0
553f976e0da38....	172.17.15.161	ProductMakeSer...	2016/12/12 7:1...	2016/12/12 7:1...	1532	39	39	0
553f976e0da38....	172.17.15.165	DataAccess	2017/2/2 2:02:00	2017/2/2 2:03:00	1408	26	26	0
553f976e0da38....	172.17.15.162	DataProcess	2017/2/10 4:44:00	2017/2/10 4:45:00	1486	33	33	0
553f976e0da38....	172.17.15.168	ImageRegisterS...	2017/1/12 5:07:00	2017/1/12 5:08:00	3562	35	35	0
553f976e0da38....	172.17.15.169	UserMgr	2017/2/16 3:57:00	2017/2/16 3:58:00	2793	7	7	0
553f976e0da38....	172.17.15.169	PosintionService	2017/1/3 3:28:00	2017/1/3 3:29:00	224	2	2	0
553f976e0da38....	172.17.15.164	UserMgr	2016/12/16 5:2...	2017/12/16 5:2...	4808	31	31	0
553f976e0da38....	172.17.15.163	DataProcess	2017/2/4 8:41:00	2017/2/4 8:42:00	2640	10	10	0
553f976e0da38....	172.17.15.167	jzbusiness	2016/12/10 1:2...	2016/12/10 1:2...	155	11	11	0
553f976e0da38....	172.17.15.161	UserMgr	2017/1/20 9:01:00	2017/1/20 9:02:00	2960	20	20	0

Figure 10. Log model result data of the service platform of IDRAS.

As can be seen from the data in Fig. 10, SaaS components of UserMgr and ProductMakeService receive 47 and 46 user requests respectively in a certain minute, but they only respond to 45 requests, and their average response time are fluctuating abnormally or fluctuating up over a time period in which the request timeouts, this indicates that the two SaaS components may be experiencing a PI. About what causes this result, the expert can combine the raw log data in the HBase database to analyze whether an exception or error has occurred in a time window in which the request timeouts. The expert can locate the reasons of SaaS software PI through tracking analysis of abnormal or error log.

The analysis results were evaluated by performance experts from IDRAS who has 8 years of experience in SaaS software performance analysis and deep knowledge of the IDRAS infrastructure. In this evaluation the experts focus on evaluating whether the result data generated by our method reflects the software is experiencing PI, the evaluation results as shown in Table II and Table III.

TABLE II. PI EVALUATION 1

PI ID: 1 Date: 2016-12-22 07:15:56 Strategy used: tracing
Manual diagnosis: Page reads/sec high on UserMgr. Cause: server restarted → cache empty so needs to be filled up.
Verification: Is real PI: Yes Diagnosis correctness: 1

TABLE III. PI EVALUATION 2

PI ID: 2 Date: 2017-01-23 06:57:56 Strategy used: tracing
Manual diagnosis: Images download timeout (9000 ms) has expired. Cause: cache I/O busy → read and write exception so needs to optimize the cache mechanism.
Verification: Is real PI: Yes Diagnosis correctness: 1

From the above two evaluation results can be seen, the results data generated by our method reflect the software is experiencing PI, it is consistent with the expert's diagnostic results. Software running cycle is long, we just choose some real-world performance problems that prove the effectiveness of our method, in the next running time, as long as the performance issues occur, we can find the performance experts to confirm, and so far our all results are corroborated with the experts.

V. CONCLUSION

From the demand and research status of SaaS software performance monitoring and analysis in cloud computing environment, this paper proposes a format standard that standardizes the SaaS software running log. Based on the standardized log, we calculate the performance metrics of SaaS software components with the aid of the service average response time calculation model and service request timeout ratio calculation model, and then obtain the log model supporting SaaS software performance analysis. In addition, this paper also presents a management framework and technology implementation of the raw log data and log model data, which supports the performance analysis of SaaS software. Finally, we demonstrate the effectiveness of our approach through a case study. The next step will improve the work of the log service module, and combine the low-level indicators to study the automatic identification of SaaS software performance issues.

ACKNOWLEDGMENT

This work is supported in part by the grants of National Key Research and Development Program of China (2016YFC1202204) and National Natural Science Foundation of China (61373038, 61672392).

REFERENCE

[1] Wei Xu. Detecting Large Scale System Problems by Mining Console Logs. PhD dissertation, UC Berkeley, 2010.

[2] Deqing Zou, Hao Qin, Hai Jin. UiLog: Improving Log-Based Fault Diagnosis by Log Analysis. Journal of Computer Science and Technology, 2016, 31(5): 1038-1052.

[3] Fu X, Ren R, Zhan J, et al. LogMaster: mining event correlations in logs of large-scale cluster systems. 2012 IEEE 31st Symposium on Reliable Distributed Systems(SRDS), 2012:71-80.

[4] Keith A. Bare, Soila Kavulya, Jiaqi Tan, et al. ASDF: An Automated, Online Framework for Diagnosing Performance Problems. Workshop on Software Architectures for Dependable Systems(WADS), 2009: 201-226.

[5] Zhiling Lan, Ziming Zheng, Yawei Li. Towards Automated Anomaly Identification in Large-Scale Systems. IEEE Transactions on Parallel and Distributed Systems(TPDS), 2010, 21(2): 174-187.

[6] Li Yu, Zhiling Lan. A Scalable, Non-Parametric Method for Detecting Performance Anomaly in Large Scale Computing. IEEE Transactions on Parallel and Distributed Systems(TPDS), 2016,27(7): 1902-1914.

[7] Aceto G, Botta A, De Donato W, et al. Cloud monitoring: A survey. Computer Networks. 2013, 57(9): 2093-2115.

[8] Mona Attariyan, Michael Chow, Jason Flinn. X-ray: Automating Root-Cause Diagnosis of Performance Anomalies in Production Software. 10th USENIX Symposium on Operating Systems Design and Implementation(OSDI), 2012: 307-320.

[9] Zhu Baojin. Design and Implementation of Log Filter System for Cloud Computing System. Hangzhou: Hangzhou Dianzi University, 2014.

[10] Xu Z, Liu Y, Mei L, et al. Semantic based representing and organizing surveillance big data using video structural description technology. Journal of Systems and Software, 2015, 102: 217-225.

[11] Liu J, Yu X, Xu Z, et al. A cloud - based taxi trace mining framework for smart city. Software: Practice and Experience, 2016.

[12] Cortellessa V, Di Marco A, Inverardi P. Model-based software performance analysis. Springer Science & Business Media, 2011. 4:10.

[13] Tao Wang, Wenbo Zhang, Jun Wei et al. Fault detection for cloud computing systems with correlation analysis. IFIP/IEEE International Symposium on Integrated Network Management(IM), 2015: 652-658.

[14] Meiyappan Nagappan. A Framework for Analyzing Software Systems Log Files. PhD thesis, NC State Uni., 2011.

[15] Holden Karau, Andy Konwinski, Patrick Wendell et al. Learning Spark: Lightning-fast data analysis[M]. CA: O'Reilly Media,Inc., 2015.

[16] Wang Zhaoyong. Research of Storage Methods for Large-scale bulk log data. Chengdu: University of Electronic Science and Technology of China, 2011

Conceptual Software Design: Algebraic Axioms for Conceptual Integrity

Iaakov Exman and Phillip Katz
Software Engineering Department
The Jerusalem College of Engineering – JCE - Azrieli
Jerusalem, Israel
iaakov@jce.ac.il phillipkatz1@gmail.com

***Abstract*—Software Conceptual Integrity has been considered a cardinal concern for design and development of software systems. But, except for verbal descriptions of desiderata, there were no formal tools to guarantee that conceptual integrity is attained. This paper proposes an axiomatic algebraic approach, based upon the assumption that the software system in each level of the system hierarchy is represented by a Modularity Matrix. This representation enables to translate propriety, orthogonality and generality, usually taken as basic conceptual integrity principles, into formal algebraic criteria. The novelty is that one can finally make Conceptual Integrity quantitative calculations. The latter are illustrated by a case study. The paper also discusses the intimate relationship between software and knowledge, which emphasizes the importance of conceptual principles for software design.**

Conceptual Integrity; software system design; Linear Software Models; Modularity Matrix; axiomatic approach; formal algebraic criteria; quantitative calculations; Software Knowledge.

I. INTRODUCTION

The central importance of Conceptual Integrity for software system design and development, first advanced by Brooks [3], is a revolutionary idea, which did not percolate yet deep enough through the practice of Software Engineering. It has two fundamental implications:

a- software is far away from the machine, be it a real or a virtual machine;

b- software understanding by human or robotic stakeholders, either a developer or a user, is at the heart of high quality design of any software system.

This paper claims that faithful translation of informal verbal principles to formal mathematical criteria is essential to widespread practical application of Conceptual Integrity ideas. Our proposal formulates algebraic axioms for Conceptual Integrity based upon the Modularity Matrix [10], an algebraic structure of the Linear Software Models.

A. The Idea of Conceptual Integrity

The idea of Conceptual Integrity was proposed by Brooks [3], [4] to deal with neat design of software, and management of teams of developers of software systems.

DOI: 10.18293/SEKE2017-148

The three principles suggested by Brooks [4] and verbally described, e.g. by Jackson and co-authors [7], [17], [18] are:

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;
3. ***Generality*** – a single function should be usable in many ways.

These principles have not been formalized, besides being explained in natural language and illustrated by examples say by some kind of informal graphical representation.

The ideas of the Conceptual Integrity principles can be summarized in terms of design modularity and simplicity. Orthogonality is a basic mechanism to obtain modularity; propriety means an optimization i.e. the fewer the functions that perform *exactly* the same tasks, the simpler the software product; generality avoids unnecessary proliferation of *similar* functions, i.e. reuse existing functions as far as possible (but see the Discussion in section VI on faithfulness of translation).

B. Linear Software Models: the Modularity Matrix

We concisely review the Modularity Matrix, a Linear Software Models' algebraic structure, representing a software system. We aim to explain the primitive terms of these models used in the axioms of this work; for details see [8], [9], [10].

Structors – the Modularity Matrix columns – are vector characterizations of software structure, generalizing classes for any hierarchical software system level. *Functionals* – the Modularity Matrix rows – are vector characterizations of software behavior, i.e. generalized functions provided by structors that may be potentially invoked. *Modules* – the blocks along the Modularity Matrix diagonal – are sub-systems, made of structor sub-sets and their respective functionals, disjoint to sub-sets corresponding to other modules. Modules give the standard Modularity Matrix its visually recognizable block-diagonality appearance.

In Linear Software Models the design goal is to achieve for a software system, the standard Modularity Matrix, in which all blocks are orthogonal. When an outlier couples modules, which are not anymore orthogonal, the system must be redesigned.

There is a two-fold relationship between a Modularity Matrix and Conceptual Integrity: a- the structors are in fact basic *concepts* of the software system; b- the immediate algebraic translation of conceptual integrity principles such as orthogonality into operations among matrix entities.

C. Paper Organization

The remaining of the paper is organized as follows. Section II refers to related work. Section III introduces the axiomatic approach. Section IV provides quantitative Conceptual Integrity criteria enabling actual calculations. Section V illustrates the approach with a case study. Section VI concludes the paper with an overall discussion.

II. Related Work

A. Applications of Conceptual Integrity

Works that mention the importance of Conceptual Integrity often make a vague statement about its meaning, e.g. Beynon et al. [2].

Kazman and Carriere [20] deal with reconstructing a software system architecture. They use *conceptual integrity* as a guideline to a meaningful architecture, viz. use a small number of components connected in regular ways, with consistent functionality assigned to these components.

Kazman [19] describes a SAAMtool, in which *Conceptual Integrity* is estimated by the number of primitive patterns that a system uses.

Clements et al. [6] think of *conceptual integrity* as the unifying theme of the system design at all levels. Similar things should be done in similar ways, with parsimonious data and control mechanisms in a system. They refer to hierarchical levels of systems, and also suggest counting mechanisms as a path to a more quantitative definition of *Conceptual Integrity*.

A recent example of the importance of Conceptual analysis for applications of Software Engineering is the work by Zambonelli [25] on key abstractions for the Internet of Things.

B. Algebraic Representations of Software Systems

In this work we refer to the Modularity Matrix [13]. Other matrices have been used for modular design.

The Laplacian matrix – see e.g. von Luxburg [21] and references therein – has been used in various applications. Exman and Sakhnini [14] have derived a Laplacian matrix containing equivalent information about a software system as the corresponding Modularity Matrix, enabling similar software design results.

The Design Structure Matrix (DSM) has been incorporated into the 'Design Rules' by Baldwin and Clark [1]. It has been applied in various contexts, including for software systems, e.g. Cai et al. [5]. An essential difference between DSM and the Modularity Matrix is linearity as a central idea of the Modularity Matrix, while DSM design quality is estimated by an external economic theory superimposed on the DSM matrix.

Conceptual lattices, analyzed within Formal Concept Analysis (FCA) were introduced in Wille [24]. There are generic overviews describing its mathematical foundations, see e.g. Ganter and Wille [16].

The equivalence between Modularity Matrices and Conceptual Lattices – see e.g. Exman and Speicher [12] – is especially relevant for the current work, since it gives a formal justification for dealing with structors as concepts.

III. An Axiomatic Approach

In this paper we choose an Axiomatic Approach for a theory of Conceptual Software Design. This kind of approach is commonly found in certain sub-fields of mathematics and physics. Why do we need axioms here?

The first justification is that we are dealing with principles that are accepted as truth without a proof. We are unable to compact the software developer experience of Brooks, as admirable as revealed in his works, and turn it into a formal argumentation. We do acknowledge that his suggested principles are largely true, and build a model based on them.

The second justification is that we are presenting here a theory combining two independent knowledge domains, which apparently have no intrinsic relationship. Again, this relationship need to be clarified and made intuitive, but cannot be proven. One domain is the field of *Software*, having its own vocabulary and syntax – say classes, functions, inheritance, etc. The other domain is a mathematical infra-structure, which here is *Linear Algebra*, having its own vocabulary and operational rules – say matrices, scalar product, eigenvectors, etc.

This is the usual way of formulating theories in science: one chooses a mathematical formulation (here Linear Algebra) judged to be the most suitable to represent and manipulate the entities of the specific scientific domain (here Software). We need to demonstrate as far as possible that the axioms in algebraic terms are a plausible and faithful translation of the intentions of Brooks' principles referring to software conceptual integrity. This kind of demonstration goes beyond the purely algebraic arguments given under the rubric of Linear Software Models in a series of papers (see e.g. Exman [8]).

The third justification is the specific contents of the proposed axioms. These have a *prescriptive* rather than descriptive nature. The axioms embody the principles of Conceptual Integrity, which are the recommended way to design software systems as argued by Brooks [4] and other authors. Again recommended and reasonable, but not formally proven.

The ultimate justification for these axioms, which are the main contribution of this work, should be provided by rigorous empirical verification in a software design laboratory.

A. Overall Characteristics of the Axioms

The three axioms of our theory of software systems design take the verbal principles behind Conceptual Integrity as mentioned above – in sub-section *A* of the Introduction section – and represent them by linear algebraic operations within the Linear Software Models. The three axioms are presented in the logical order of the matrix manipulation needed to obtain the best system design.

B. First Axiom: Propriety

Remember the verbal formulation of Propriety: software systems should just have all the appropriate functions as demanded by its requirements, but no more than those. Its meaning is therefore to avoid superfluous functions.

1st Software Design Axiom: Propriety
For any hierarchical level of a given Software System represented by its Modularity Matrix, all its structors should be linearly independent, and all its functionals should be linearly independent.

Linear independence (of both structors and functionals) obtains exactly the desired *propriety*. Identical columns/rows in the Modularity Matrix should be eliminated, leaving just one instance of each column/row kind. If a few columns/rows are respectively linear dependent, their number should be reduced to the bare essentials, i.e. the minimal number of independent vectors.

An important algebraic consequence of the Propriety demands is that standard Modularity Matrices for any given software system are square, as shown in [10].

C. Second Axiom: Orthogonality

The verbal formulation of orthogonality is that any individual functions should be independent of one another. Its meaning is thus, no overlap at all between these functions. This formulation with "independence" appears to imply linear independence, but the latter is already achieved with the 1st axiom. Orthogonality in algebra is a stronger demand than linear independence, deserving a separate axiom.

2nd Software Design Axiom: Orthogonality
For any hierarchical level of a given Software System represented by its Modularity Matrix, all the structors/functionals belonging to a given module should be respectively orthogonal to all the structors/functionals belonging to any other module.

Orthogonality as defined in Linear Algebra, i.e. the scalar product of two vectors is zero, is indeed a stronger demand than linear independence, viz. it demands zero overlap between vectors.

Orthogonality of software system modules is made visible by reordering columns and rows in the Modularity Matrix, displaying the disjoint sets of structors and their respective functionals, the blocks along the matrix diagonal, viz. showing block-diagonality.

It seems quite clear that Brooks and other authors concerned with conceptual integrity have used the term "orthogonality" inspired by its mathematical meaning. So, it is natural to keep using exactly the same term with its perfect meaning correspondence. The specific contribution of this work is to determine which entities – which vectors – have their orthogonality calculated.

D. Third Axiom: Generality

The verbal formulation of Generality is that a function should be usable in many ways. Our interpretation is that similar functions should not be replicated in different modules, but refined into a single generic form and then possibly invoked from other modules, if necessary. In other words, in any hierarchy level of the software system, modules should be cohesive enough – functions related to one another within a module – while enabling *composition* with other modules, even from different hierarchical levels.

3rd Software Design Axiom: Generality
For any hierarchical level of a given Software System represented by its Modularity Matrix, all its modules should display high cohesion. If a module does not display high cohesion it should be split, maximizing the number of modules in the Matrix.

High cohesion in terms of modules of the Modularity Matrix means low sparsity, i.e. low numbers of zero-valued matrix elements relatively to the module size.

If a module has low cohesion it should be split by means of a procedure using eigenvectors of the Modularity Matrix (see e.g. [11]).

E. Application of Conceptual Software Axioms

The importance and practical usage of the above axioms is to obtain software system modules actually displaying Conceptual Integrity. To this end, one just needs to apply on the software system Modularity Matrix the standard operations found in Linear Software Models. These are:

1. For *propriety* – eliminate linear dependencies respectively of matrix columns/rows;
2. For *orthogonality* – reorder matrix columns/rows to obtain block-diagonality, where blocks are modules;
3. For *generality* – eliminate outliers, thereby reducing block sizes, to avoid low module cohesion, increasing the number of modules.

IV. Quantitative Criteria Calculations

Besides the formal axioms, the practical significant contribution of this work is to formulate explicit equations to *calculate quantitative criteria* to measure conceptual integrity, one criterion fitting each of the above axioms. These equations are combined to calculate criteria for a whole software system.

All the quantities involved in the calculations of Conceptual Integrity are normalized. Normalization means that these quantities are independent of the vector or matrix sizes, i.e. one divides results by relevant entity sizes.

A. Equations for Conceptual Integrity Criteria

We assume that all elements of the Modularity Matrix, and therefore of its modules, structors and functionals are non-negative. The normalized criteria are given as follows:

1. *Propriety* – Linear Dependence within a module is evaluated by equation (1), in which **r** is the rank and **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 which is obtained when **r** equals **c**.

$$\textbf{\textit{Propriety}} = 1 - ((c - r)/c) \qquad (1)$$

2. *Orthogonality* – assume a pair of vectors ***u*** and ***v*** where each of them is normalized [23], i.e. all their elements are divided by the length of the respective vector. Their Orthogonality criterion is evaluated by equation (2), where $(u \bullet v)$ is the vectors' scalar product. Orthogonality has a value between zero and the maximal value 1 obtained for zero scalar product.

$$\textbf{\textit{Orthogonality}} = 1 - (u \bullet v) \qquad (2)$$

3. *Generality* – for each module of a software system the generality criterion is given by equation (3), the *Cohesion* of the module, where *Sparsity* is the ratio between zero-valued matrix elements and the total number of matrix elements of the module. This generality expression – the module cohesion – has a value between zero and 1 and is maximal when the Sparsity is minimal.

$$\textbf{\textit{Cohesion}} = (1 - \textbf{\textit{Sparsity}}) \qquad (3)$$

B. Systems' Conceptual Integrity Calculations

Software system calculations, based upon the above equations, should be done for the whole set of Modularity Matrix modules to obtain the combined conceptual integrity criterion for the respective software system.

For instance, the orthogonality criterion of a system with ***n*** modules, where ***k*** modules are mutually orthogonal and the remaining ***n-k*** are not orthogonal, is calculated by equation (4). The measured orthogonality ***m(i)*** for each non-orthogonal module ***i*** is obtained by repeated application of equation (2). Thus,

$$\textbf{\textit{System\_Orthogonality}} = \left(k + \sum_{i}^{n-k} m(i)\right) / n \qquad (4)$$

V. ILLUSTRATION BY A CASE STUDY

A system calculation is here illustrated by a simple case study, shown in Fig. 1. It has 5 modules, with one outlier coupling between two modules (viz. modules 2 and 3), and 3 mutually orthogonal modules (viz. modules 1, 4 and 5).

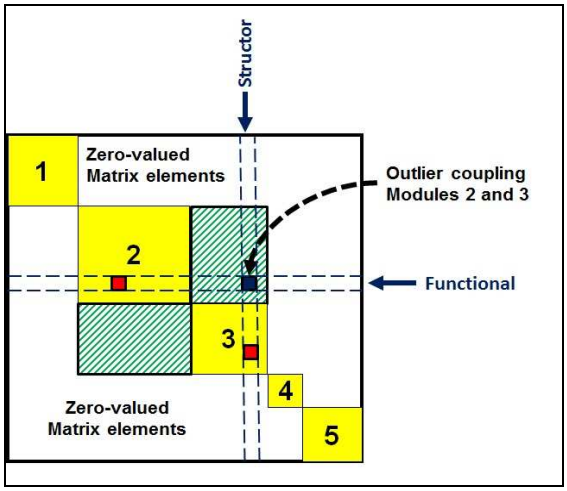


Figure 1. Schematic generic Modularity Matrix with outlier – It has 5 modules (in yellow), three of them mutually orthogonal (modules 1, 4 and 5). Modules 2 and 3 are coupled by the outlier (in blue). At least one matrix element (in red) in the coupled modules is shared by a Functional and a Structor containing the outlier. The hatched areas (in green) and the coupled modules form a larger module, too sparse to be a legitimate module, as most hatched areas matrix elements are zero-valued, except for the outlier.

The outlier couples modules 2 and 3 as it has a Functional (row) shared with module 2 and a Structor (column) shared with module 3. As seen in Fig. 1, the shared Functional has at least one non-zero matrix element in module 2. Similarly, the shared Structor has at least one non-zero matrix element in module 3. Thus, the scalar products – in equation (2) above – of the Functional and Structor containing the outlier with at least one respective Functional of module 2 and with at least one respective Structor of module 3 will be greater than zero.

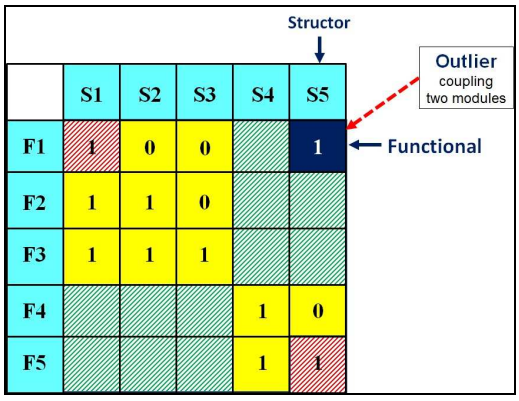


Figure 2. Specific Modularity Sub-Matrix with outlier – It is equivalent to zooming into (yellow) modules 2 and 3 of Fig. 1, whose hatched (red) elements are coupled by a (dark blue) outlier. The top-left module size is 3*3 and the bottom-right size is 2*2. The hatched areas (in green) outside the modules have all zero-valued elements (ommitted for simplicity), except the outlier.

The eigenvectors' procedure [11] obtaining Modularity Matrix modules, results in a larger module containing modules 2 and 3 and the outlier element. But, cohesion calculation of

this larger module, applying equation (3) above, shows that this larger module is too sparse to be a legitimate module. One effectively breaks it into modules 2 and 3, leaving the matrix element outside these smaller modules as an outlier.

Now let us calculate the orthogonality given by equation (4) for the software system with non-orthogonal modules shown in Fig. 2. The outlier has a functional F1 in common with the top-left module. F1 has non-zero scalar products with two other functionals (F2 and F3) of the same module. Thus, by applying twice equation (2) the average measured orthogonality ***m(i)*** for the top-left module is 0.55. In addition, the outlier has a structor S5 in common with the bottom-right module. S5 has a non-zero scalar product with the structor S4. Thus the measured orthogonality ***m(i)*** for the bottom-right module is 0.5. Therefore, the System Orthogonality calculated value is 0.81.

VI. DISCUSSION

A. The Relation between Functions and Concepts

The verbal Conceptual Integrity principles, in sub-section *A* of this paper's Introduction, are expressed in terms of functions, instead of *concepts*, as it was pointed out by Jackson and co-authors [7]. A Conceptual Integrity exposition is expected to focus on concepts. One needs to explain the role of functions. Here this is done in two steps.

In the first step, we reiterate that in the Object Oriented approach to software design, classes and their generalization *structors* stand for abstract types, i.e. actually natural language concepts. Functionals, a generalization of functions, enable the vectorial representation of Structors. In other words, they are the behavioral characterization of the respective concepts.

The second step is a reminder that the Modularity Matrix with its Structors and Functionals have been shown by Exman and Speicher [12] to be equivalent to a *conceptual lattice*. Lattices, within Formal Concept Analysis [16], are algebraic structures linking concepts of a particular domain. Thus, from this point of view, functions are relevant to conceptual analysis.

B. The Number of Conceptual Integrity Axioms

The number of Conceptual Integrity axioms is an open issue. The very formalization of the axioms should facilitate answering this question, as axioms within the formalism should display coherence. Almost paradoxically, we should apply Conceptual Integrity to decide about the number of its axioms.

It should be kept in mind, that the exact number of axioms and their specific contents are dependent on software design and development technologies. A substantial amount of automatic development and less human influence, say by deep learning techniques, would probably keep the algebraic basis for the axioms, but change specific contents.

De Rosso and Jackson [7] proposed a 4th verbal principle called *Consistency* enunciating that actions behave in a similar way irrespective of the states in which they are invoked. They did not include it among the Conceptual Integrity principles, as it seems a user interface issue rather than a deeper conceptual issue. Such verbal principle could generate a 4th axiom.

Additional axioms are certainly conceivable. But, are we sure that the minimal set is indeed three? This issue is dealt with in the next sub-section.

C. Faithfullness of Translation

The use of a mathematical formalism is never purely arbitrary. We have chosen Linear Algebra to represent software design since our analysis refers to linearly dependent information, such as conceptual dependencies, function replications and common functionalities. On the other hand, it is well known that certain science domains may be equally well represented by quite different mathematical tools. For instance, quantum mechanics was expressed through a differential equation by Schroedinger, and through matrices by Heisenberg, and it was necessary to demonstrate their non-obvious equivalence.

The issue of faithful translation of the verbal principles into formal axioms appears to be relevant to the exact number of axioms. In terms of linear algebra coherence, Orthogonality and Propriety seem to be completely justified, as they explicitly refer to linear algebra operations.

Generality, on the other hand, may raise some controversy and certainly deserves a deeper discussion. First, cohesion is a software rather than an algebraic concept, even though sparsity in equation (3) is expressed in terms of matrix notions.

Generality does not appear to be an intrinsic property of a given design of a software system. Through the representation of high-level abstraction ideas behind the software, in a broad and open sense, it rather seems to prepare the ground for *future extensions* and modifications.

D. Abstraction as a Deeper Motivation for Conceptual Integrity

The deeper motivation for the importance of Conceptual Integrity in software design and development can be inferred from the software history along time in the last fifty years. One perceives the continuous increase of the highest abstraction level from which one starts software design, with concomitant distancing away from machines, be they real or virtual.

We cautiously distinguish two kinds of abstraction hierarchies. One kind is the software system proper hierarchical levels, each level with its own Modularity Matrix. This hierarchy refers to the whole system, sub-systems, sub-sub-systems, down to indivisible components chosen by software engineers. Besides the modularity partition of the software system, it also characterizes abstraction levels. Modules in each level have a specific conceptual identity, which is more and more general as one goes up the hierarchy.

The other kind of abstraction levels is that of the implementation languages hierarchy, from the lowest machine level up to the human natural language. This hierarchy in bottom-up direction approximately passes from assembly, to high-level programming languages, to models such as UML, and finally to natural language concepts.

Higher abstraction levels imply easier grasping of the software system design by humans. Thus Conceptual Integrity is not a constraint artificially imposed from the outside, but an inherent and comprehensible quality serving human stakeholders, both the software developers and its end-users.

E. Software and Knowledge

Since this paper is presented to a conference on Software and Knowledge Engineering, it is worthwhile to stress the intimate relationship between Software and Knowledge. On the one hand, Conceptual Integrity is of cardinal importance to software system design and development.

On the other hand, concepts are first class entities in structures used to define and impart meaning to the vocabulary of a certain domain. Typical such structures are the conceptual lattices, referred above in sub-section *A* of this Discussion and their near cousins, the domain and application ontologies, which are the subject of Knowledge engineering.

F. Pragmatic Considerations

In order to apply in practice the theory exposed in this paper one needs a software tool such as Modulaser [15], enabling generation and analysis of Modularity Matrices for software systems from compiled code. One can also easily generate a Matrix from UML class diagrams, in which classes stand for structors and independent methods stand for functionals. It should be stressed that such analysis points out to coupling problems deserving system redesign, while leaving the actual solution to considerations of the software engineer.

Finally, one observes that Modularity Matrices solve a software system composition problem, somewhat similar to Petri Nets. However, the latter focuses on distributed systems, and Modularity Matrices are not specialized for these systems.

G. Main Contribution

There are two main contributions of this paper. First, the formalization of the principles behind Conceptual Integrity in an axiomatic fashion, provide a deeper justification for the algebraic operations found in the Linear Software Models.

Second, the formulation of equations finally enables calculations of quantitative criteria for Conceptual Integrity.

REFERENCES

[1] C.Y. Baldwin and K.B. Clark, *Design Rules*, Vol. I. The Power of Modularity, MIT Press, Cambridge, MA, USA, 2000.

[2] W.M. Beynon, R.C. Boyatt and Z.E. Chan, "Intuition in Software Development Revisited", in Proc. of 20th Annual Psychology of Programming Interest Group Conference, Lancaster University, UK, 2008.

[3] F.P. Brooks, *The Mythical Man-Month – Essays in Software Engineering* – Anniversary Edition, Addison-Wesley, Boston, MA, USA, 1995.

[4] F.P. Brooks, *The Design of Design: Essays from a Computer Scientist*, Addison-Wesley, Boston, MA, USA, 2010.

[5] Y. Cai and K.J. Sullivan, "Modularity Analysis of Logical Design Models", in *Proc. 21st IEEE/ACM Int. Conf. Automated Software Eng. ASE'06*, pp. 91-102, Tokyo, Japan, 2006.

[6] P. Clements, R. Kazman and M. Klein, *Evaluating Software Architecture: Methods and Case Studies*. Addison-Wesley, Boston, MA, USA, 2001.

[7] 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.

[8] I. Exman, "Linear Software Models", Proc. GTSE 1st SEMAT Workshop on a General Theory of Software Engineering, KTH Royal Institute of Technology, Stockholm, Sweden, 2012a. http://semat.org/wp-content/uploads/2012/10/GTSE_2012_Proceedings.pdf.

[9] I. Exman, "Linear Software Models", video presentation of paper [8] at GTSE 2012, KTH, Stockholm, Sweden, 2012b. Web site: http://www.youtube.com/watch?v=EJfzArH8-ls.

[10] 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.

[11] 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: http://dx.doi.org/10.1142/S0218194015500308

[12] I. Exman and D. Speicher, "Linear Software Models: Equivalence of the Modularity Matrix to its Modularity Lattice", in Proc. 10th ICSOFT'2015 Int. Conference on Software Technology, pp. 109-116, ScitePress, Portugal, 2015. DOI: 10.5220/0005557701090116

[13] I. Exman, "Linear Software Models: An Algebraic Theory of Software Composition", in Proc. 28th Int. Conf. on Software Engineering and Knowledge Engineering, Keynote Abstract, KSI Research, Redwood City, CA, USA, 2016.

[14] I. Exman and R. Sakhnini, "Linear Software Models: Modularity Analysis by the Laplacian Matrix", in Proc. 11th ICSOFT'2016 Int. Conference on Software Technology, Volume 2, pp. 100-108, ScitePress, Portugal, 2016. DOI: 10.5220/0005985601000108

[15] I. Exman and P. Katz, "Modulaser: A Tool for Conceptual Analysis of Software Systems", in Proc. SKY 2016, 7th Int. Workshop on Software Knowledge, pp. 19-26, ScitePress, Portugal, 2016.

[16] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*, Springer-Verlag, Berlin, Germany, 1998.

[17] 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

[18] 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.

[19] R. Kazman, "Tool Support for Architecture Analysis and Design", in ISAW'96 Proc. 2nd Int. Software Architecture Workshop, pp. 94-97, ACM, New York, NY, USA, 1996. DOI: 10.1145/243327.243618

[20] 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.

[21] U. von Luxburg, "A Tutorial on Spectral Clustering", *Statistics and Computing*, 17 (4), pp. 395-416, 2007. DOI: 10.1007/s11222-007-9033-z

[22] W. Reisig, *Understanding Petri Nets*, Springer, Berlin, 2013.

[23] E.W. Weisstein, "Normalized Vector" From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/NormalizedVector.html

[24] 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.

[25] F. Zambonelli, "Key Abstractions for IoT-Oriented Software Engineering", IEEE Software, pp. 38-45, 2017. DOI: 10.1109/MS.2017.3

A Method to Analyze High Level Petri Nets using SPIN Model Checker

Dewan Mohammad Moksedul Alam
School of Computing and Information Sciences
Florida International University
Miami, Florida 33199, USA
dalam004@fiu.edu

Xudong He
School of Computing and Information Sciences
Florida International University
Miami, Florida 33199, USA
hex@cis.fiu.edu

Abstract— **High level Petri nets (HLPNs) are a formal method for studying concurrent and distributed systems and have been widely used in many application domains. However, their strong expressive power hinds their analyzability. In this paper, we present a new transformational analysis method for analyzing a special class of HLPNs – predicate transition (PrT) nets. This method extends and improves our prior results by covering more PrT net features including full first order logic formulas and exploring additional alternative translation schemes. This new analysis method is supported by a tool chain – front end PIPE+ for creating and simulating PrT nets and back end SPIN for model checking safety and liveness properties. We have applied this method to two benchmark systems used in annual Petri net model checking contest 2015. We discuss several properties and show the detailed model checking results of two properties in one system.**

Keywords— formal methods; high level Petri nets; temporal logic; model checking; SPIN

I. INTRODUCTION

High level Petri nets (HLPNs) are a powerful formal method for modeling concurrent and distributed systems. HLPNs provide a graphical representation of systems to make them easier to understand. HLPNs offer strong expressive power through rich data abstraction, and algebraic expressions and logic formulas for defining system functionality. Furthermore, the dynamic semantics of HLPNs supports model level simulation. As a result, HLPNs are being used widely in system modeling in many application domains.

However, due to the expressive power of HLPNs, they are difficult to analyze. This is especially critical when we need to study the safety and liveness properties of highly dependable systems modeled in HLPNs. In recent years, a variety of new analysis techniques based on model checking have been proposed to analyze high level Petri nets in addition to traditional techniques such as simulation. These new analysis techniques fall into two general categories: (1) developing tailored model checking algorithms for particular types of HLPNs, or (2) leveraging existing general model checkers through model translation where a HLPN is transformed into an equivalent form suitable for the target model checker.

Although tailored model checkers for HLPNs may take advantages of the unique features of the underlying HLPNs and perform specific optimizations to improve performance, they often lack the full-fledged features provided by the well-known general model checkers and are not easily adaptable. Furthermore, there are no convincing experiments yet to show tailored model checkers have performance advantages over using general model checkers. Several tools supporting direct model checking of HLPNs participated in the annual Petri net model checking contest, but had very limited success.

An alternative way of analyzing HLPNs leverages existing well-known general purpose model checkers. A translation method is needed to generate a state transition system from a given HLPN. Model translations have been used and explored for many years [1]. The key of a translation method is to ensure the behavioral equivalence of the source model and target model. It is quite straightforward to obtain a state transition system from a given HLPN with the loss of true concurrency. Fortunately, safety and liveness properties are preserved from all known Petri net semantics – causal (partial order), maximal concurrency, interleaving set, and interleaving. In our prior work ([2], [3], [4]), we developed methods to translate a particular type of HLPNs – predicate transition nets (PrT nets) into the input models of several well-known model checkers including SMV [5] and SPIN [6] with various limitations. In this paper, we present a new and more general method to translate PrT nets to SPIN. This method includes the following new contributions: (1) eliminating many limitations such as no set variables on arc label in our prior work, (2) supporting advanced features including quantifiers in full first order logic formulas, (3) combining transition enabling testing and firing to improve performance, and (4) offering an additional process based translation scheme. These new features have been implemented in our enhanced PIPE+ environment previously reported in [7]. We demonstrate the application of our method through two benchmark systems used in the annual model checking contest of Petri net tools 2015 [8].

II. PREDICATE TRANSITION NETS

In the following sections, we give a brief introduction to PrT nets. More detailed definitions of PrT nets can be found in [9].

A. The Syntax and Static Semantics of PrT Nets

A PrT net is a tuple *(N, Spec, ins)* where $N = (P, T, F)$ is a net structure, in which (1) P and T are disjoint nonempty finite sets (the sets of places and transitions of N respectively), (2) $F \subseteq P \times T \cup T \times P$ is a flow relation (the arcs of N). *Spec = (S,*

DOI reference number: 10.18293/SEKE2017-162

Op, Eq) is the underlying algebraic specification, and consists of a signature $\Sigma = (S, Op)$ and a set *Eq* of Σ-equations. Signature Σ includes a set of sorts *S* and a family $Op = (Op_{s1,s2\ldots sn;s})$ of sorted operations for $s1, \ldots, sn, s \in S$. For $s \in S$, we use Con_s to denote *Op;s* (the 0-ary operation of sort *s*), i.e. the set of constant symbols of sort *s*. The Σ-equations in *Eq* define the meanings and properties of operations in *Op*. *Spec* is a meta-language to define the tokens, labels, and constraints of a PrT net. Tokens of a PrT net are ground terms of signature Σ, written $MCon_S$. The set of labels is denoted by $Label_S(X)$ (*X* is the set of sorted variables disjoint with *Op*). Each label can be a simple variable or a set expression of the form $\{x_1, x_2, \ldots, x_n\}$. Constraints of a PrT net are a subset of first order logic formulas (where the domains of quantifiers are finite and any free variable in a constraint appears in the label of some connecting arc of the transition). The subset of first order logical formulas contains the Σ-terms of sort *Bool* over *X*, denoted as $Term_{(Op;Bool)}(X)$.

$Ins = (\varphi, L, R, M_0)$ is a net inscription that associates a net element in *N* with its denotation in *Spec*. $\varphi : P \to \wp(S)$ is the data definition of *N* and associates each place p in *P* with a subset of sorts in *S*. $L : F \to Label_S(X)$ is a sort-respecting labeling of PrT net. We use the following abbreviation in the following definitions: $\bar{L}(x, y) = L(x, y)$ if $(x, y) \in F$ or $\bar{L}(x, y) = \emptyset$ if $(x, y) \notin F$. $R : T \to Term_{(Op;Bool)}(X)$ is a well-defined constraining mapping, which associates each transition *t* in *T* with a first order logic formula defined in the underlying algebraic specification. Furthermore, the constraint of a transition defines the meaning of the transition. We use *var(t)* to denote variables appearing in *R(t)*. $M_0 : P \to MCons_S$ is a sort-respecting initial marking. The initial marking assigns a multiset of tokens to each place *p* in *P*.

A Σ-algebra of *Spec* provides interpretations for the sorts and operations in *Spec*, which includes familiar sorts such as integer, Boolean, string, tuple, and set as well as their relevant operations and equations. In our tool environment, the Σ-algebra is instantiated with a subset of Java data types and their associated operations and laws.

B. The Dynamic Semantics of PrT Nets

The dynamic semantics of PrT nets are defined on the concept of markings (states) that are mappings from places to tokens. A transition *t* in *T* is *enabled* in a marking *M* if there are tokens in its input places that satisfy its precondition defined in constraint *R*(*t*). Formally, the enabling condition can be defined as: $\forall p \in P. (\bar{L}(p, t) : \alpha \subseteq M(p) \wedge R(t) : \alpha)$, where α is an instantiation of arc variables with tokens in p. An enabled transition *t* can *fire* by removing tokens from its input places and adding tokens to its output places according to the post condition defined in constraint *R*(*t*), which results in a new marking *M'*. Formally, transition firing can be defined as: $M'(p) = M(p) \cup \bar{L}(t, p) : \alpha - \bar{L}(p, t) : \alpha$ for all $p \in P$, where α is an instantiation satisfying $R(t) : \alpha$. 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. The dynamic semantics (behavior) of a PrT net is the set of all possible transition firing sequences starting from the initial marking. The essential dynamic semantics regarding the translation method are discussed later.

III. THE MODEL CHECKER SPIN

Promela is the underlying modeling language for SPIN to describe a system model. An operational model in Promela contains one or more *process*es, zero or more *variables*, zero or more *message channels* and a *semantics engine* [6].

Processes are the central construct in a Promela model to define system behaviors, and are defined using *proctype* declaration.

Message channels are used to model the transfer of data from one state to another, which can store a finite number of messages as declared. Apart from storing data, channels provide a wide range of features to model message passing in a clean and efficient way. Basically, they are FIFO queue but can also be used for random accesses. When a new message is sent to a channel, it is added to the end. When an attempt is made to retrieve a message, it always returns a message in front. It is also possible to query a channel for a specific message. By default, channel removes messages as they are retrieved. However, it is also possible to get a message without removing it. Some basic channel usages are given below.

Table 1- Basic usage of message channel

```
1.  chan qname = [8] of {int, short}
2.  chan qname = [8] of {s_type}
3.  qname!10, 20
4.  qname?x, y
5.  qname?x, eval(y)
6.  qname?[x, eval(y)]
7.  qname??[x, eval(y)]
8.  len(qname)
```

In the above example, lines 1 and 2 show the declaration of channels with basic datatypes and structured datatypes respectively. Line 3 sends message to the channel. Lines 4-7 show different ways to retrieve messages. Lines 4, 5 and 6 retrieve first message. Lines 4 and 5 also remove the message after retrieving. Lines 5 and 6 check whether the second element of the first message matches the value currently hold in y. Line 7 searches for a match anywhere in the channel. Lines 6 and 7 retrieve a message without removing it (poll operation). Line 8 returns the number of messages in the channel.

Promela provides non-determinism by default. The case selection and looping shown in Table 2 both have similar constructs. Each case is marked with a guarded statement (started with ::). If there are more than one case, then the sequence of statements will be selected for which the guarded statement is executable. If there are more than one such statement, then one of those will be chosen non-deterministically. If no such statement present, then the block is exited. A loop tries to find one executable guarded statement each time it completes execution. Promela also has a *for* loop construct, which is only to be used to iterate through channels.

Table 2 - Examples of control constructs

Case selection	Looping
if :: case 1 :: case 2 :: case 3 fi	do :: case 1 :: case 2 :: case 3 od

The next important thing is inline functions. Although Promela does not have general function concept, but supports inline functions as macros. We take advantage of in-line functions to structure our translated code.

IV. TRANSLATION OF PRT NETS TO PROMELA

A complete translation needs to translate all the components of a PrT net to equivalent constructs in the target language. Our new translation method eliminates many limitations in our prior work ([3], [4]) and supports many advanced features such as complex data structures and first order logic formulas. We have also combined transition enabling testing and firing to improve model checking performance. Furthermore, we have implemented an additional new process based translation scheme, which has a major impact in checking some safety and liveness properties that are otherwise not checkable using non-deterministic selection of in-line function based transition translation. We discuss the main translation rules as well as identified problems in the following sections.

A. Translation of Places

In PrT nets, *places* basically store information called tokens and thus represent the states of a net. Furthermore, PrT nets are a data flow computation model. State changes occur through token movements. The place concept perfectly matches the *channel* concept in Promela. The built-in functions of *channel* make it easy to query specific tokens for checking transition enabling condition and to add/remove tokens for transition firing. Each place in a PrT net is thus translated into a channel with the same type and the tokens of in the place are translated into messages stored in that channel in the initialization. Thus, for all $p \in P$, we have the following two lines as declarations.

```
#define bound_p const
chan    place_p = [bound_p] of {type_p}
```

Here, *type_p* is a data type in Promela resulted from the translation of the data type of place *p*, which will be discussed in the translation of data types.

B. Translation of Transitions

Transitions are the core components of dynamic semantics of a net and play the most important role in the execution as discussed in a prior section. The execution of a transition has two parts: testing its enabling condition and then firing it if enabled.

The translation of transitions constitutes the operational model in Promela. Each transition $t \in T$ is translated using the algorithm shown in Table 3. The techniques adopted for evaluating precondition and post-condition are discussed in the translation of transition constraints.

C. Translation of Arcs

Arcs are not translated directly but arc labels are important in determining the input and output variables of transitions during the translation of transition constraints (i.e. precondition and post-condition).

Table 3 – A general algorithm to translate a transition

```
inline check_is_enabled_and_fire_t() {
  for all combinations of values the input
  variables to t can take, do the following
    if preconditions are satisfied then
        compute postconditions
        compute new marking
        return;
  done
}
```

D. Translation of Data Definition

The data definition of places are multisets over the set of the sorts. These multisets are translated as structured types in Promela, where each sort in the data type definition is represented as a field in the structure.

PIPE+ supports only two basic sorts – string and number. These are used to define more complex data types for places through Cartesian product. To translate these complex data types, we first need to determine the corresponding representations of the sorts. Promela does not support strings, we choose *mtype* in Promela to represent strings. Each string token is treated as an enum constant in *mtype*. Promela does not support real numbers, but provides three types to support integers: `int`, `short`, and `byte`, in which the latter two help to reduce the state space. We use `short` as the default implementation and let a modeler select other choices during translation. For example, a place *p* with data type, $\varphi(p)$ = <*String, Number, String*> is translated into a Promela structured data type shown in Table 4.

Table 4 – An example of translation of data type definition

```
typedef struct type_p {
     mtype field1;
     short field2;
     mtype field3;
}
```

E. Translation of Transition Constraints

The constraint of a transition $t \in T$ is defined using a first order logic formula, which specifies the relationships among input and output variables of *t*. The constraint has two parts – precondition only involving input variables and the post-condition containing output variables.

1) Translation of operators

PIPE+ supports a variety or algebraic, relational, logical, and set operators. A PrT net uses a subset of these operators to define their behavior. Table 5 shows the supported operators in PIPE+.

Table 5 – Supported operators in PIPE+

Category	Operators	Data type
Algebraic	+, -, *, /, %	number
Relational	=, ≠, <, ≤, >, ≥	number,string
Logical	∧, ∨, ¬, →,↔	bool
Set	∈, ∉, ∪, \, |, :, ·	
Quantifiers	∀, ∃, ∄	

Most of these operators have their usual meanings. Some are overloaded, such as = operator. When = is used in a precondition, it is a comparison. When '=' is used in a post-condition, it acts as an assignment.

2) Translation of Preconditions

Each logic expression without quantifiers in a PrT net is translated into an equivalent form supported in Promela by using the corresponding operators. For example, expression $(x[1] \geq 3.5 \wedge y[1] \neq 10) \rightarrow z[1] = 2 * x[1] + 1$ is translated to the equivalent expression in Promela shown in Table 6.

Table 6 – An example of translation of expression

```
!(x.field1 >= 3.5 && y.field1 != 10)|| z.field1
== (2*x.field1+1)
```

However, the set operators and quantifiers in Table 5 do not have equivalent representations. The translation of quantifiers is given later. The translation of set operations is partially completed and is not further discussed in this paper since they are rarely used based on our experiences.

3) Translation of Post-conditions

Translation of post-conditions involve evaluation of the expressions and assignments of the values to the designated output variables. Alike preconditions, the evaluation is almost the same. Only difference is that, the evaluated values are assigned to the designated output variables. These output variables will be used as messages to corresponding output channels during new marking generation.

F. Translation of the Initial Marking

An initial marking defines the initial state of a PrT net. The structured tokens of each place are simply translated into send statements to the corresponding channel.

G. Translation of Quantifiers

PIPE+ supports two types of quantifiers – universal (∀) and existential (∃). A first order logic formula with quantifiers is called a quantified formula. Quantifiers are essentially used as a part of preconditions for seeking desirable input tokens. Thus we focus on how to translate these formulas in checking transition enabling conditions.

We identify different types of quantified formulas based on their structures, which are important in correctly handling the quantifiers. (1) Is a quantified formula within another formula (possibly another quantified formula), (2) Does a quantified formula contain global variables (arc annotations and those from their outer quantifiers), (3) Can a quantified formula be nicely separated into a conjunction of precondition and post-condition, (4) Does a quantified contain another quantified formula within it and so on.

For each quantifier, an inline function implementing an algorithm like what shown in Table 3 is generated. The difference is that, for an existential quantifier, as soon as the first combination of tokens satisfying the enabling condition, the loop exits. And for a universal quantifier, as soon as the first combination of tokens violates the enabling condition, the loop stops. The in-line functions for quantifiers have the following benefits: (1) easy management of the nested quantifiers. It is only a matter of calling the inline functions for the nested quantifier, (2) declaration, definition and scope management of the local and global variables can be resolved easily. The underlying contract of inline functions provided by Promela takes care of the above tasks.

Only care needed is to make sure unique names are used in these inline functions and the loop control variables have the correct names.

The in-line functions for quantifiers are called inside the innermost for loop before the *if* block in the corresponding precondition evaluation function shown in Table 3. This way, we can ensure the availability of the arc variables.

H. Putting Pieces Together

So far we have discussed the techniques of translating each component of PrT nets into Promela's constructs. To complete the translation, we need to provide an overall execution structure based on the dynamic semantics of PrT nets. There are two essential ways to select a transition firing in Promela – (1) as a passive in-line function to be non-deterministically selected in a Do loop within a centralized process. As long as there is an enabled transition, the loop continues; or (2) as an active *process*, which will be selected for execution by the SPIN runtime if the enabling condition is true. The first strategy was used in our prior work, where a transition's enabling condition checking and firing were executed separately, which not only had the slow performance but also could result in incorrect result due to conflict. We resolve this problem in our new implementation by combining transition enabling checking with firing, which has also improved performance. Furthermore, we have also implemented the second strategy. This new translation strategy results in much better model checking results in detecting violations of some safety properties and is important to check liveness properties when weak fairness assumptions are needed.

I. Translation Correctness

The correctness of the translation method covers the completeness and consistency. Completeness measures whether all features of PrT nets are translated into Promela. Our current translation covers all PrT features except a few set operations, which will be implemented in near future. Consistency refers to the equivalence between the dynamic behavior of a PrT net and the dynamic behavior of the translated Promela program. Of course, we ignore the concurrency transition firings in PrT nets that do not affect the satisfiability of safety and liveness properties that are essentially state based. Therefore, we only need to compare interleaved executions between a PrT net and the translated Promela program. Each interleaved execution starts from the initial marking and continues by firing one enabled transition at a time. Using an induction proof principle, we can only show (1) the initial marking is translated correctly, (2) each transition is translated correctly, i.e. the enabling condition and firing result are translated correctly, and (3) each enabled transition will be selected to fire. Step (1) is trivial true in our translation method. Step (2) is arguably true based on our careful design and extensive testing. Step (3) is true for both our overall model execution strategies discussed in the previous section. However, the formal proof of a general translation method is not easy, which is the reason that few compilers have been formally verified.

V. EXPERINEMT RESULTS

To evaluate our translation method, we have used two benchmark systems, Bridge and SafeBus, from the annual Petri net model checking contest 2015 [8]. These are the only high level Petri net models available in the contest, which were defined using colored Petri nets (a type of high level Petri nets). These systems are redefined using PrT nets in PIPE+. We only present and describe the Bridge model in this paper due to the page limit. The PrT models of the Bridge system and the SafeBus system with different parameters in xml format and the generated Promela models with properties in pml format as well as SPIN model checking results can be found at https://bitbucket.org/ptnet/pipe/.

The bridge system represents a system of a single lane bridge with an automatic controller for controlling the two-way through traffic. The model proposed contains three control parameters (V, P, N), where V is the number of vehicles on each side of the bridge trying to get to the other side, P is the maximum number of vehicles allowed on the bridge, and N is the maximum number of vehicles from the same side allowed to pass in a row. A PrT model of the system with parameters (4, 5, 2) is shown in Fig. 1. The net inscriptions are shown in Table 7 and 8.

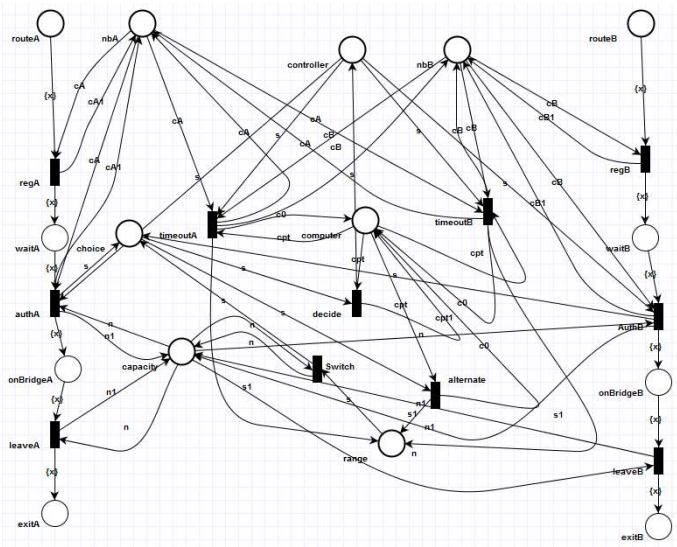


Fig. 1 – A PrT net model of the Bridge system

We have run simulations of PrT model in PIPE+ and simulations of the translated Promela model in SPIN. Both simulations ended with all vehicles crossed the bridge successfully. Although simulations in PIPE+ are much faster. The properties checked include: (1) the number of vehicles on the bridge never exceeds the maximal allowed (parameter P), (2) all the vehicles on the bridge must move in the same direction, (3) all the vehicles eventually cross the bridge, and (4) it cannot happen when some vehicles are at starting point and others have crossed the bridge. Properties (1) to (3) are the desirable ones –

Table 7 – Data type definitions of the places

```
φ(routeA)=φ(waitA)=φ(exitA)=φ(onBridgeA)= ℘(number)
φ(routeB)=φ(waitB)=φ(exitB)=φ(onBridgeB)= ℘(number)
φ(nbA)=φ(nbB)=φ(timeoutA)=φ(timeoutB)= number
φ(controller)=φ(computer)=φ(capacity)= number
φ(choice)=φ(range)= number
```

Table 8 – Constraint definitions of the transitions

```
R(regA) = (cA1=cA+1)
R(authA)=((s=1∧n>0∧cA>0)∧(cA1=cA-1∧n1=n-1))
R(leaveA) = n1=n+1
R(timeoutA) = (s=1∧cA=0∧cB>0∧s1=2∧c0=0)
R(regB) = (cB1=cB+1)
R(authB)= ((s=2∧cB>0∧n>0)∧(cB1=cB-1∧n1=n-1))
R(leaveB)= n1=n+1
R(timeoutB) = ( s=2∧cB=0∧cA>0∧s1=1∧c0=0)
R(decide) =  (cpt<5∧cpt1=cpt+1)
R(alternate) = ((cpt=5∧c0=0∧s=1∧s1=2)∨
               (cpt=5∧c0=0∧s=2∧s1=1))
R(switch) = (n=5)
```

expected to hold, but property (4) is undesirable – expected to fail. These properties are specified using LTL expressions as (1)[]!((len(place_onBridgeA) > P || len(place_onBridgeB)) > P), where P is a constant based on the given model, (2) []!(len(place_onBridgeA)>0&&len(place_onBridgeB)>0), (3) for all x, [](routeA(x) -> <>exitA(x)) and a similar formula for the opposite direction B, and (4) []!(len(place_routeA)>0 && len(place_exitA)>0).

In these experiments, properties (1) to (3) on all the Promela models up to (20,10,10) have been checked successfully without any violation, however due to the complexity of these models, SPIN quickly reached the allotted memory bound of 2048 Mbyte. Thus, only bounded state space is searched. The reported none violation can be false positive. Since all verification runs of the above properties in a particular model resulted in similar results except slight time differences since the state space explored is the same bounded by the allotted memory. A summary of SPIN model checking results of property (1) in the translated Promela models are shown in Tables 9 and 10. Table 9 shows the results where the transitions are translated as passive in-line functions to be non-deterministically selected to execute. Table 10 shows the results where transitions are translated as active processes. These experiments show that the first approach explores much smaller state space and is very fast than the second one. However, the second approach is effective in detecting violations in some safety properties, which found a counter example of property (4) in 47 milliseconds; while the first approach failed in finding a counter example in bounded search space as shown in Table 11. The second approach is also needed when a liveness property depends on weak fairness assumption since SPIN runtime environment will enforce fairness for processes.

Table 9 – Model checking results of property (1) using approach 1

Parameters (V, P, N)	State Transitions	Atomic steps	Search Depth	Time (seconds)
(4, 5, 2)	1479955	2437508	2823	2.99
(10,10,10)	1677016	2775407	5754	3.62
(20,10,10)	3670366	5990664	12730	10

Table 10 – Model checking results of property (1) using approach 2

Parameters (V, P, N)	State Transitions	Atomic steps	Search Depth	Time (seconds)
(4, 5, 2)	26172925	2.27E+08	4307	36.2
(10,10,10)	40727761	3.63E+08	7591	76.1
(20,10,10)	25392316	2.31E+08	14739	64

Table 11 – Checking results of property (4) with parameter (4, 5, 2)

Approach	States	Depth	Time	Result
1	9323528	2841	23.3	No-error
2	828	2351	0.047	Error

SPIN was not able to run large models due to state space explosion problem, which also happened the Petri model checking contest where none of the participating tools could verify the above high level Petri net models.

VI. Related work

During the past two decades, SPIN has been used as the analysis engine of many tool development efforts. SPIN has been used to model check programs. One of the most prominent work is Java PathFinder [11], which is a prototype translator from Java to Promela. Another work involves translation from C code to Promela [12]. SPIN has also been used as for model check specifications and designs. In [13], a formal approach was proposed to verify web service orchestration by translating the web service business process execution language (WS-BPEL) to Promela. Several studies [14, 15] attempted to formally verify the UML system models using SPIN by translating UML models to Promela.

Formal verification of Petri net models using SPIN have also been explored in the past several years. In [16], a simple technique was proposed to translate low level Petri nets to Promela. Several other similar techniques were proposed in the literature to translate low level Petri nets, but few works dealt with high level Petri nets. In our prior work [3, 4, 10], we proposed several techniques to translate PrT nets to Promela and implemented some of them with some restrictions in our tool environment PIPE+. Those concepts and their implementations suffer from some limitations – (1) those were not generic enough to support a wide range of system models, (2) did not support advanced features like quantifiers in full first order logic formulas, (3) suffered from some performance bottlenecks, (4) the translation scheme was rigid having no way to tweak the translation process, etc. are worth mentioning. Our new translation described in this paper eliminates these problems. The model translation feature in PIPE+ is fully automatic. Once a model is translated, we can explore powerful features of SPIN to verify the constraints and properties of the model using iSPIN graphical user interface.

VII. Conclusions

In this paper, we presented a method to translate PrT nets into a Promela programs. This new translation method supports many advanced features of PrT nets and provides a new execution scheme. The method is implemented in our tool environment PIPE+ and is completely automatic. Once a model is translated to Promela program, we can leverage SPIN's model checking capability to analyze system properties of PrT net models. Currently, we are doing more experiments to test SPIN's model checking capabilities, which will provide insights for model construction and specific PrT features to use. Furthermore, we want to develop a strategy to combine PrT's simulation capabilities and SPIN's model checking capabilities in analyzing different types of system properties when model checking may not be feasible. We will also utilize modeling knowledge to guide the translation, for example, mapping bounded integers in a PrT model to byte type in Promela, which may improve checking performance or make some property checking feasible. We will also fully implement the set operations in case some models require them. Our tool is open source and is available at https://bitbucket.org/ptnet/pipe/.

Acknowledgement

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. We thank anonymous reviewers' comments for improving the presentation.

References

[1] S. Katz and O. Grumberg, "A Framework for Translating Models and Specifications," in Proceedings of the Third International Conference on Integrated Formal Methods, pp. 145–164, Springer-Verlag, 2002.

[2] X. He, H. Yu, T. Shi, J. Ding, and Y. Deng, "Formally analyzing software architectural specifications using SAM," J. Syst. Softw., vol. 71, no. 1-2, pp. 11–29, 2004.

[3] G. Argote-Garcia, P. J. Clarke, X. He, Y. Fu, and L. Shi, "A Formal Approach for Translating a SAM Architecture to PROMELA," in SEKE, pp. 440–447, 2008.

[4] Lily Chang and Xudong He: "A Methodology to Analyze Multi-Agent Systems Modeled in High Level Petri Nets", International Journal of Software Engineering and Knowledge Engineering – IJSEKE, vol.25, no.7, 2015, 1199-1235.

[5] "The SMV System", http://www.cs.cmu.edu/~modelcheck/smv.html.

[6] G. Holzmann, "The Spin Model Checker: Primer and Reference Manual," Addison-Wesley Professional, 2003.

[7] Su Liu and Xudong He: "PIPE+Verifier - A Tool for Analyzing High Level Petri Nets", Proc. of the 27th International Conference on Software Engineering and Knowledge Engineering (SEKE15), Pittsburgh, July 6 – 8, 2015.

[8] "Model checking contest @ Petri Nets", http://mcc.lip6.fr/models.php.s.

[9] 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.

[10] Su Liu, Reng Zeng, Zhuo Sun, and Xudong He, "SAMAT - A Tool for Software Architecture Modeling and Analysis," Proc. of the 24th International Conference on Software Engineering and Knowledge Engineering, San Francisco, CA, pp. 352-358, July 2012.

[11] Havelund K: "Java PathFinder A Translator from Java to Promela". In: Dams D., Gerth R., Leue S., Massink M. (eds) Theoretical and Practical Aspects of SPIN Model Checking. SPIN 1999. Lecture Notes in Computer Science, vol 1680. Springer, Berlin, Heidelberg

[12] K. Jiang and B. Jonsson: "Using SPIN to Model Check Concurrent Algorithms, using a translation from C to Promela". In: Proc. 2nd Swedish Workshop on Multi-Core Computing, Uppsala, Sweden: Department of Information Technology, Uppsala University , 2009, 67-69.

[13] H. H. Kacem, W. Sellami, A. H. Kacem. "A Formal Approach for the Validation of Web Service Orchestrations". 2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. June 2012. doi: https://doi.org/10.1109/WETICE.2012.53.

[14] L. Ji, J. Ma and Z. Shan: "Research on Model Checking Technology of UML". 2012 International Conference in Computer Science and Service System, p.p. 2337 - 2340, 2012.

[15] Y. Yamada and K. Wasaki: "Automatic generation of SPIN model checking code from UML activity diagram and its application to Web application design", The 7th International Conference on Digital Content, Multimedia Technology and its Applications, p.p. 139 - 144. 2011.

[16] G. C. Gannod and S. Gupta: "An automated tool for analyzing Petri nets using Spin", Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001), pages 404-407, 2001.

Algebraic Formalization and Verification of PKMv3 Protocol using Maude

Jia She, Xiaoran Zhu, Min Zhang(✉)
Shanghai Key Laboratory of Trustworthy Computing,
MoE International Joint Lab of Trustworthy Software, ECNU, 200062, China
melodyspencer@126.com, zhangmin@sei.ecnu.edu.cn

***Abstract*—PKMv3 is the third version of Privacy and Key Management protocol, which plays an important role by providing key distribution and security access control in IEEE802.16m, the standard of Worldwide Interoperability for Microwave Access. The protocol should be guaranteed safe in terms of confidentiality, authentication and integrity. In this paper, we develop an executable formal specification of PKMv3 in an algebraic language called Maude and verify safety properties of the protocol using state exploration and LTL model checking in Maude. Unlike existing approaches, we consider the behaviors of intruders and time feature in our verification and verify both safety properties and time-related properties of the protocol.**

I. Introduction

IEEE802.16m [4] is the newer generation of WiMAX, a communication standard for long-distance high-speed transmission of wireless data. In IEEE802.16m, PKMv3 protocol is defined for key management and used to control security access to the network by establishing encryption connections. Based on its first two generations, PKMv3 improve message hierarchical protection and its encryption algorithm. In addition, PKMv3 support Extensible Authentication Protocol (EAP) method and Cipher-based Message Authentication Code (CMAC), which largely reduce the possibility of attack.

The mobile WiMAX network faces more threats than traditional wireless networks. Security of the network must be guaranteed. Many efforts have been done to verify security properties of PVM protocol. In [8], BAN logic is used to analyze the key management protocols of the previous two generations, and the defects of one-way authentication in PKMv1 and the interspersed attacks in PKMv2 are pointed out. In [5], Scyther (an automated protocol testing tool) is used to implement the formal analysis of PKMv2 protocol to verify the defects of confidentiality, authenticity and integrity in the protocol. In [9], the PKMv2 protocol is described by the Casper protocol modeling tool and the output of the process communication is analyzed by FDR tool. It is found that the intruder can intercept the message and replay the attack. In [7], the security flaws of authentication and authorization have been in WiMAX, such as the possible DoS attack, and the defects of authentication and key space. Raju et. al. also use CasperFDR tool to model the process of transmitting plaintext messages between base station, mobile station and relay station in PKMv3 protocol, and get the attack model of stealing secret keys by analyzing outputs [6]. Zhu et al consider the time characteristics of PKMv3 protocol key lifecycle and model them using Promela language and use DT-Spin model checker to verify the protocol liveness, key period and message consistency [10].

In this work, we formalize PKMv3 protocol in an algebraic language called Maude [1] with considering both the behaviors of intruders and the time feature of the protocol. Six properties including three safety properties and three time-related properties are verified using the searching and LTL model checking functions provided by Maude. The verification results coincide with those in existing works [5], [7], [6], [10]. Besides that, we find the integrity vulnerability of PKMv3 by verification and propose a solution on the basis of the verification result.

II. PKMv3 Protocol

PKMv3 protocol provides security distribution mechanism of key material between the server called *base station* (BS) and the client called *subscriber station* (SS). More precisely, it generates traffic encryption keys (TEK) for stations to encrypt the messages they exchange. The generation of TEK consists of three procedures which are called *AK authentication*, *SA negotiation* and *TEK transmission*. Each procedure is achieved by exchanging key information between SS and BS. After a TEK is generated, secret keys i.e., *AK* and *TEK*, may expire because their lifetime is limited. After they expire reauthorization is needed. Figure 1 depicts the whole process of key generation and reauthorization. In this section, we give the details of three procedures and reauthorization.

A. AK Authentication

By this phase, SS and BS establish mutual authentication and SS receives a Pairwise Master Key (PMK) sent by BS. PMK is used by BS and SS to generate Authorization

This material is based upon work supported by National Natural Science Foundation of China (NSFC) project: No. 61502171.
DOI reference number: 10.18293/SEKE2017-061

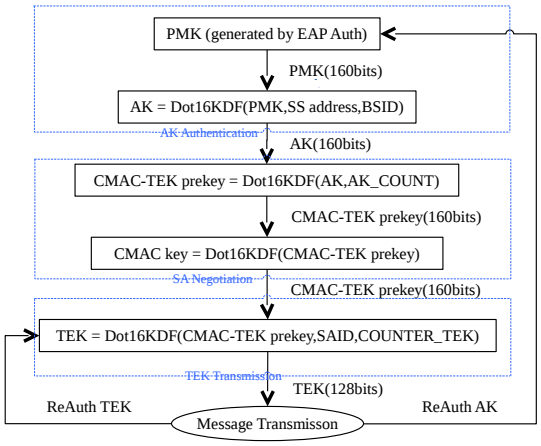


Fig. 1. The process of key generation and reauthorization

Key (AK), which are used in the subsequent process. This phase consists of the following three message exchanges:

Msg1.	$SS \rightarrow BS$:	$Cert_{SS}\|N_S\|Capabilities\|SIG_{SS}$
Msg2.	$BS \rightarrow SS$:	$Cert_{BS}\|N_S\|N_B\|ENC_{SS}(PMK)\|$ $PMK\_SN\|SIG_{BS}$
Msg3.	$SS \rightarrow BS$:	$N_B\|SS\_Address\|CheckSum$

Firstly, SS sends BS an auth-request message, which is encrypted with the private key of SS as a digital signature, SIG_{SS}. The message includes a digital certificate of SS, a nonce N_S generated by SS and the encryption capabilities that SS supports. On receiving the message, BS decrypts it with the public key of SS, and then sends an auth-response message encrypted with the private key of BS. This message includes a certificate of BS, the nonce N_S received from SS, a nonce N_B generated by BS, a PMK encrypted with SS's public key, and the sequence number PMK_{SN} of PMK. SS decrypts the message and authenticates the legitimacy of the BS's digital certificate. Finally, SS sends BS an auth-success message containing the address of SS, with which an AK can be generated.

B. SA Negotiation

By the phase, SS and BS negotiate the security association (SA) through three message exchanges. SA is an information set that is required for secure communication. The three message exchanges are shown below:

Msg4.	$BS \rightarrow SS$:	$N_B\|PMK\_SN\|AKID\|CMAC$
Msg5.	$SS \rightarrow BS$:	$N_B\|N_SPMK\_SN\|AKID\|SNP\|$ $Security\ Capabilities\|CMAC$
Msg6.	$BS \rightarrow SS$:	$N_S\|PMK\_SN\|SAID\|SNP\|CMAC$

Before message exchange, SS and BS generate CMAC keys using AK separately. CMAC keys are used to calculate CMAC digest. BS firstly sends SS a key agreement message, including nonce N_B generated in this round, PMK_{SN}, and identifier $AKID$ of AK. In particular, BS generates a CMAC digest with the three arguments and sends along with the message. On receiving the message, SS calculate a CMAC digest using its CMAC key and compares it with the received one. If they are equal, SS sends BS the second key agreement message, including N_B, PMK_{SN}, AKID, security negotiation parameter (SNP), and newly generated nonce N_S and CMAC digest. Once BS checks the validity of the CMAC digest, it sends SS the identifier of SA (SAID) and other information. SAID is used to identify different security levels of security association. At this point, SS and BS have negotiated the necessary information required for secure communication.

C. TEK Transmission

By this phase, SS and BS generates TEK after three message exchanges, as shown below.

Msg7.	$SS \rightarrow BS$:	$SAID\|PMK\_SN\|TEK\ Refresh$ $Flag\|CMAC$
Msg8a.	$BS \rightarrow SS$:	$SAID\|PMK\_SN\|Counter\_TEK$ $EKS\|CMAC$
Msg8b.	$BS \rightarrow SS$:	$SAID\|Wrong\_Code\|CMAC$

SS first sends BS a TEK-Request message, including SAID, PMK_SN, TEK refresh flag and CMAC digest. After checking the correctness of CMAC digest and SAID, BS sends SS a request confirm message TEK-Reply, including SAID, PMK_SN, the counter of TEK (Counter_TEK), the encryption key sequence (EKS), and the CMAC digest. Counter_TEK is used to generate TEK together with SAID and the CMAC-TEK prekey that is derived by AK, and EKS is used to distinguish consecutive TEKs. If the TEK request fails, BS sends SS a request invalid message TEK-Invalid, including SAID, error code and the CMAC digest. In this case, SS needs to re-send TEK-Request message to BS.

D. Reauthorization

After the whole process of the three procedures, BS and SS establish a security connection for the transmission of communication messages encrypted by TEK. As mentioned above, because both AK and TEK have limited lifetime BS and SS have to be re-authorized to keep liveness and safety when the lifetime of the keys expire.

When AK's lifetime expires, SS needs to send an Auth-request message to BS to request a new AK. It ends the current process and restart from AK authentication, as shown in Figure 1. Because TEK's lifetime is embedded in AK's, when TEK becomes invalid but it corresponding AK is still alive, SS only needs to send TEK-request message to BS to start the process of TEK transmission.

III. Maude Nutshell

Maude is an algebraic specification language and also an efficient rewriting engine [1]. The underlying rewriting logic of Maude is a logic of concurrent changes well-suited to formalize states and concurrent computations [2].

A. Formalization in Maude

A system is formalized in Maude as a rewrite theory i.e., $(\Sigma, E \cup A, R)$, consisting of a signature Σ, a collection E of (possibly conditional) equations and memberships defined on Σ, a collection A of equational attributes, and a set R of rewrite rules. The two-tuple $(\Sigma, E \cup A)$ is called a membership equational theory, which specifies the "statics" of a system, i.e., the algebraic structure of the set of system states, and the rules in R specify the "dynamics" of the system, i.e., all the possible transitions that the system can perform. In Maude, a system state is usually represented as an algebraic inductive structure which can be a tuple, a *soup* of components, or even an object. Thanks to the inductive representation, Maude can naturally specify both finite-state and infinite-state systems. Transitions among the states are formalized by rewrite rules. Given two states υ_1 and υ_2, let t_{υ_1} and t_{υ_2} be their corresponding terms. There is one-step transition from υ_1 to υ_2 if there is a rewrite rule r such that t_{υ_1} can be rewritten into t_{υ_2} by applying r once.

Finally, we briefly summarize the syntax of Maude (see the work [1] for more details). Sorts and subsort relations are declared by the keywords `sorts` and `subsort`, respectively. Operators are declared with the `op` keyword in the form: `op` f `:` $s_1 \ldots s_n$`->` s, where $s_i (i = 1, \ldots, n)$ and s are sorts. Maude allows for user-defined mixfix operators and uses underbars in operators to indicate each of the argument positions. An equation in Maude is declared in the form of `eq` t `=` t', where t and t' are two terms of the same sort. An equation can be conditional, which is declared by the keyword `ceq` and ended with the keyword `if`, followed by a conjunction of conditions. A rewrite rule in Maude is declared in the form of `rl [label]:` t `=>` t'. A conditional rewrite rule is declared by the keywords `crl` and `if` with a conjunction of rewrite conditions.

B. Formal analysis in Maude

Maude programs are executable. For this feature Maude provides multiple formal analysis methods, mainly including simulation, reachability analysis, and model checking to formally analyze systems.

Simulation is achieved using Maude's `rewrite` command, which repeatedly applies the rewrite rules to transform a given term step by step. The transformation process simulates one behavior of the specified system.

Maude provides a `search` function to explore the reachable state space of specified systems. The `search` function can be used to verify invariant properties of systems. An invariant property states that something bad should never happen. We verify an invariant property by specifying the negation of the property as the condition in `search` command and using Maude to find if there are solutions. A solution can be interpreted as a counterexample of the property. If the reachable state space is infinite or finite but too large to be explored due to time or memory limitations, a bound on the searching depth is needed, and thereafter properties are partially verified in such cases.

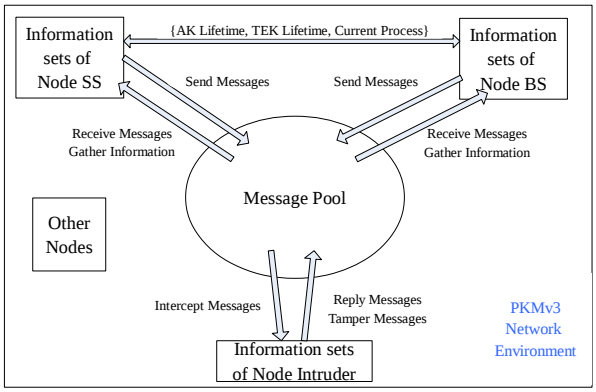


Fig. 2. Configuration of PKMv3 network

Maude also provides an efficient LTL model checker [3] to verify LTL properties of systems. Given a Maude specification and a set of atomic propositions, the model checker takes an initial state and an LTL formula and returns true or a counterexample. A condition of doing model checking in Maude is that the set of states that are reachable from the given initial state must be finite.

IV. Formal Modeling of PKMv3 Protocol

This paper treats PKMv3 protocol and its environment as a dynamic system. In the network, there are stations e.g., SS, BS and other intruder, and message transmissions, as depicted in Figure 2. We assume the correctness of all the encryption algorithms used in the protocol and one BS communicates with only one SS at a time.

A. Definition of basic data types

In Maude we define abstract data types to formalize the objects in the network such as stations, messages, lifetime and nonce. Due to space limitation, we only explain the main data types as examples.

We declare a sort `Station` for stations and three subsorts `Ss`, `Bs` and `Intruder` of it for BS, SS, and intruder respectively. Three constructors `ss`, `bs` and `intruder` are defined to construct stations for SS, BS and intruder with a string as their argument.

```
sorts Station Ss Bs Intruder .
subsorts Ss Bs Intruder < Station .
op ss : String -> Ss [ctor] .
op bs : String -> Bs [ctor] .
op intru : String -> Intruder [ctor] .
```

We use random numbers to model nonce, as formalized by the following Maude codes. A random number can be a seed or a successor of an existing one. A nonce consists of a station which generates it, a station to which the nonce is to be sent, and a random number.

```
sorts Rand Nonce .
ops seed seed0 seed1 : -> Rand [ctor] .
op  next : Rand        -> Rand [ctor] .
op nonce : Station Station Rand -> Nonce [ctor] .
```

Keys and certificates of stations are formalized likewise. As defined below, constructor pubkey (resp. prikey) constructs a public (resp. secret) key, macaddr constructs a MAC address of a station, and cert constructs a certificate with a station, a MAC address and a public key.

```
sorts Pubkey Prikey MacAddr Cert .
op pubkey  : Station -> Pubkey  [ctor] .
op prikey  : Station -> Prikey  [ctor] .
op macaddr : Station -> MacAddr [ctor] .
op cert : Station MacAddr Pubkey -> Cert [ctor] .
```

B. Formalization of system states

A system state of PKMv3 protocol consists of a set of nodes, a pool of messages and the connection time between SS and BS.

A message is formalized as a triple, consisting of source, destination and a collection of contents such as certificates, keys and nonce. Message and contents are defined in Maude as follows:

```
sorts Content Contents Message .
subsorts Key Cert Nonce ... < Content < Contents.
op _,_ : Content Content -> Content [assoc comm].
op msgnil : -> Message [ctor] .
op from_to_send_: Station Station Content ->
    Message .
```

The mixfix operator _,_ represent the union of two collection of contents, msgnil represents an empty message, and from_to_send_ constructs a message with a source station, a destination station and a collection of contents.

A node consists of a station and a collection of contents. The duration time of a connection consists of remaining lifetimes of AK and TEK and a flag to indicate to which process the communication is proceeding. We use natural numbers to represent the lifetime of AK and TEK, and formalize duration time as a triple. The Maude definition of nodes and connection time is given below:

```
sorts Node LTime Process ConnTime .
subsort Nat < LTime .
op  node[_]:_ : Station Content -> Node .
ops pak psa ptek pmt : -> Process .
op  {_,_,_} : LTime LTime Process -> ConnTime .
```

Constructors pak, psa, ptek and pmt of sort Process respectively represent the processes AK authorization, SA negotiation, TEK exchange and message transmission. Lifetime of secret keys is defined as sort LTime with predefined sort Nat. The connection time is defined by sort ConnTime and represented using the constructor {_,_,_}. It is worth mentioning that before AK (resp. TEK) is generated, the time in the triple represents the AK (resp. TEK) grace time, i.e., the time that is allowed before AK (resp. TEK) is generated [10].

As defined below, sort States is declared for system states, and Node, Message and ConnTime are declared as its subsorts. A state is composed of nodes, messages and connection time by the constructor __.

```
sorts States .
subsorts Message Node ConnTime < States .
op __ : States States -> States [assoc comm] .
```

TABLE I
VARIABLES USED IN THE SPECIFICATION

Variables	Sorts	Descriptions
A, B, C	Stations	SS, BS, Intruder
R1, R2, R0	Rand	Random number
NC_X(X=A,B,C)	Nonce	Nonce sent by Station
CERT_X(X=A,B,C)	Cert	Certificate of Station
PMKSN	Pmksn	Key sequence of PMK
AKID	AkId	Identifier of AK
AKCOUNT	AkCount	Counter of AK
MSG2	AuthResponse	Ak-response message
MSG5	SAResponse	SA-response message
CMAC1,CMAC2	Cmac	CMAC digest
CMACKEY	Cmackey	Cmac key
SAID	SaId	Identifier of SA
FLAG	TRFlag	TEK refresh flag
C_TEK	CounterTek	Counter of TEK
AKLT,TEKLT	LTime	Life time of AK,TEK
PS	Process	Process of PKMv3
CS1,CS2	Content	Message contents

C. Formalization of message exchanges

We formalize the eight message exchanges explained in Section 2 by rewrite rules in Maude. There are totally 22 rewrite rules defined. We take three of them as examples. Table 1 shows all the variables that are used in rewrite rules as well as their data types and descriptions.

The first one specifies the behavior of sending an auth-request message from SS to BS for AK authentication.

```
rl [SendAuthRequestMsgReal] :(node[A]: R0,CERT_A)
(node[B]: R1,CERT_B) (msgnil) {AKLT,TEKLT,pak} =>
(node[A]: next(R0),CERT_A,nonce(A,B,R0))
(node[B]: R1,CERT_B)
(from A to B send sencrypt(authrequest(CERT_A,
    nonce(A,B,R0),capa(A)),prikey(A)))
{sd(AKLT,tt),TEKLT,pak} .
```

The term pak in the rule indicates that the behavior occurs at the AK Authentication process. The right-hand side of the rule says after the behavior a message is put into the network. The message consists of a certificate CERT_A, a nonce generated by station A and the encryption capabilities capa(A) of station A. It is encrypted by the private key prikey(A) of A and sent to B. After sending the message, AK grace time AKLT is decreased by a fixed number tt of time units which can be customized initially.

The second rewrite rule describes the behavior of BS receiving SA-response message from SS for SA negotiation process. The condition of the behavior is that there is an SA-response message sent to BS in the network. In the message, the AKID and nonce N_B should be same as the ones that B holds, and attached CMAC digest CMAC2 is the same as the one calculated using the information in MSG5 and CMACKEY. The rewrite rule is defined as follows:

```
crl [RecvSAResponseReal] :
(node[B]:NC_B,AKID,CMACKEY,CS1)
(from A to B send (MSG5,CMAC2)){AKLT,TEKLT,ps}=>
(node[B]:NC_B,AKID,CMACKEY,getsc(MSG5),getnon52(
MSG5),getsnp5(MSG5),CS1)(msgnil){AKLT,TEKLT,psa}
if getakid5(MSG5) == AKID /\
   equals(getnon51(MSG5),NC_B) == true /\
   equalm(cmac(MSG5,CMACKEY),CMAC2) .
```

The last rewrite rule specifies key reauthorization. Key reauthorization occurs when lifetime of AK AKLT is no less than a predefined value tt of time units and TEK is less than tt in message transmission procedure, as defined by the condition part of the following rewrite rule.

```
crl [CircleTEKMsgReal] :
(node[A]:PMKSN,CMACKEY,SAID,FLAG,TEK,C_TEK,CS1)
(node[B]:PMKSN,CMACKEY,SAID,FLAG,TEK,C_TEK,CS2)
{AKLT,TEKLT,PS} (msgnil) =>
(node[A]: PMKSN,CMACKEY,SAID,flag(1),CS1)
(node[B]: PMKSN,CMACKEY,SAID,CS2 )
(from A to B send(tekrequest(SAID,PMKSN,flag(0)),
cmac(tekrequest(SAID,PMKSN,flag(1)),CMACKEY)))
{sd(AKLT,tt),sd(tgt,tt),ptek}
if AKLT >= tt /\ TEKLT < tt .
```

The right-hand side of the rule means that by reauthorization A and B delete all the data gathered in TEK exchange such as FLAG, TEK and C_TEK first, and A generates new TEK refresh flag flag(1) and sends B a TEK-request message containing SAID received in SA negotiation, PMK_SN and TEK refresh flag and CMAC digest.

D. Formalization of intruders

Intruder is a part of the network environment. We assume intruders have three capabilities: (1) participating in the protocol communication with the same capacity of normal station; (2) eavesdropping on the messages in the network; and (3) replaying received messages and tampering with message contents. Intruders may intervene in each procedure of the protocol and hinder the normal communication of SS and BS.

We formalize the behavior of intruder as rewrite rules in the same way as we formalize those of normal stations. Due to space limitation, we only discuss two as examples.

The following rule formalizes the process of intruder C intercepting auth-response message in AK authentication. When B sends A auth-response message MSG2, C can intercept this message from message pool and encrypt its digital signature using the public key of B.

```
crl [RecvAuthResponseMsgFake] :
(node[C]: R2,CERT_C,CERT_A,NC_A)
(from B to A send sencrypt(MSG2,PRI)) =>
(node[C]: R2,CERT_C,CERT_A,NC_A,getcert2(MSG2),
    getnon22(MSG2),decrypt(getcipher(MSG2),prikey
    (C)),getpmksn2(MSG2)) (msgnil)
if equals(getnon21(MSG2),NC_A) == true /\
   getpub(getcert2(sdecrypt(sencrypt(MSG2,PRI_B),
       pubkey(B)))) == (B) .
```

The condition says that if the certificate of B and N_B are correct, intruder C retrieves corresponding data from MSG2 and tries to decrypt PMK with its own secret key.

The second rewrite rule specifies the behavior of intruder C sending fake auth-confirm message in AK authentication process. After eavesdropping communication between SS and BS, intruder has gathered necessary information including nonce N_B, PMK_SN and address of station A. Hence, C can forge auth-confirm message with N_B and fake address ssaddr(C) and then disguise that SS sends BS this message.

```
crl [SendAuthConfirmMsgChange] :
(node[B]: AKCOUNT,PMKSN,CS1)
(node[C]: NC_B,PMKSN,ssaddr(A),CS2) (msgnil)
{AKLT,TEKLT,PS} =>
(node[B]: AKCOUNT,PMKSN,CS1)
(node[C]: NC_B,PMKSN,ssaddr(A),CS2)
(from A to B send(authconfirm(NC_B,ssaddr(C)),
    checksum(authconfirm(NC_B,ssaddr(C)),iv)))
{sd(AKLT,tt),TEKLT,PS}
if AKLT >= tt /\ not(ssaddr(C) inc CS1) .
```

The condition says that the message can be sent if the lifetime of AK is no less than transmission time tt and B has not received auth-confirm message yet. In addition, intruder can reply auth-confirm message to B as the identity of station A, without changing original contents of auth-confirm message which sent by A.

V. Formal Verification of PKMv3's Properties

With the specification, we verify both safety properties and time-related properties by the searching function and Maude LTL model checker.

A. Definition of initial states

An initial state should be provided for verification. We assume that in the initial state there is a mobile station, a base station and an intruder, which have their own digital certificates. The message pool is empty. Message transmission time tt is set 4, and the grace time of AK agt and TEK tgt 60. We use init to denote the initial state of PKMv3 network, which is defined as follows:

```
op init : -> States .
eq init = (msgnil) {agt,tgt,pak} (node[ss("a")]:
    seed,cert(ss("a"),macaddr(ss("a")),pubkey( ss
    ("a"))))(node[bs("b")]: seed1,cert(bs("b"),
    macaddr(bs("b")),pubkey(bs("b"))))(node[intru
    ("c")]: seed0,cert(intru("c"),macaddr(intru("
    c")),pubkey(intru("c")))) .
```

B. Formalization of properties

We consider six properties of PKMv3, including three safety properties and three time-related properties.

The three safety properties are called *confidentiality*, *authentication* and *integrity*. By confidentiality it means that intruders can never obtain PMKs that are used by two normal stations. Without PMK intruders cannot calculate AK and participate in the subsequent process even if it has gathered other important key material. Authentication is another important property of the protocol. That is, even if there are attacks from intruders in the network, normal stations should still be able to verify each other's identity and carry on communication. Integrity property says that messages transmitted between the honest subjects should not occur transmission error or be tampered with, or at least wrong message data can be detected. In Maude, safety properties can be verified using search command and do not need to specify separately.

The three time-related properties are *succession*, *reauthorization*, and *key freshness*. Succession means that the four procedures should take place in turn. Reauthorization

says that when AK and TEK expire, authorization needs to be re-established immediately. Key freshness says that transmitting those messages with expired AK and TEK is not allowed. The three properties can be specified as the following LTL formulas in Maude.

```
(eap-auth U nego-sa) /\ (<> nego-sa U exch-tek)
    /\ (<> exch-tek U msg-trans)) .
(<> msg-trans U exch-tek) /\ (<> msg-trans U eap-
    auth) ) .
[](~ inva-msg ) .
```

In the above formula, `eap-auth`, `nego-sa`, `msg-trans` and `exch-tek` are atomic propositions which are true in those states where the process tag is respectively `pak`, `psa`, `ptek` and `pmt`, and `inva-msg` is an atomic proposition which is true in those states where the lifetime of current AK or TEK is less than zero. The symbols `U`, `<>` and `[]` represent the temporal connectors **U**, **F** and **G** in LTL.

C. Formal verification and result analysis

We use searching function to verify the safety properties and Maude LTL model checker to verify the time-related properties of the protocol. For instance, the following search command is used to find a state where an intruder obtain a PMK that is not supposed to belong to it.

```
search init =>* (S:States) (node[intru("c")]:
pmk(ss("a"),bs("b"),nonce(ss("a"),bs("b"),seed),
nonce(bs("b"),ss("a"),seed1),algo),C:Contents).
```

Maude returns `no solution`, which means that the confidentiality holds. The other two safety properties can be verified likewise by searching. For model checking, one only needs to call `modelCheck(init,`F`)` with `init` the predefined initial state and F an LTL formula.

The verification results of the six properties are shown in Table II. For time-related properties, the succession and reauthorization of PKMv3 protocol can be satisfied, which means that the procedures of PKMv3 and period lifetime of secret keys are correct. Meanwhile, messages are always transmitted with valid secret keys which guarantee key freshness. For the safety properties, confidentiality and authentication can also be satisfied by our Maude model. The intruder can never obtain the PMKs and other secret keys that are not supposed to belong to it. SS and BS can always authenticate each other's identity. The verification results coincide with those in the work [5], [7], [6], [10].

Another finding in our verification is that the integrity of messages can not be guaranteed. Maude returns a case that BS receives a fake auth-confirm message with incorrect address of SS, and then generates invalid AK. The reason is that auth-confirm message is generated using common checksum algorithm, but intruder can tamper with this message and generate corresponding checksum. In this case, BS acknowledge that the received message does not have transmission error, but can not verify whether the message is tampered with or not by intruder.

The integrity vulnerability of PKMv3 can be improved by using AK to generate the checksum of auth-confirm message. More precisely, the modified message is described as: $SS \rightarrow BS$: $N_B|SS\_Address|AK(N_B|SS\_Address)$. BS generates AK with received SS_Address, and then compares the calculated checksum with the one that received. Verification results after the refinement show that the integrity of message transmission is satisfied.

TABLE II
VERIFICATION RESULTS OF THE SIX PROPERTIES

Property	Method	Rewrite	Time	Result
Succession	Model checking	388	2ms	√
Reauthorization	Model checking	264	4ms	√
Key freshness	Model checking	1199	2ms	√
Confidentiality	Searching	1843	44ms	√
Authentication	Searching	1843	112ms	√
Integrity	Searching	420	4ms	×

VI. CONCLUSION

We have presented an algebraic approach to formal modeling of PKMv3 protocol and verification of its six safety properties and time-related properties using Maude. In our model, we consider both the behaviors of intruders and the time feature of the protocol. Verification results show that our model of PKMv3 can satisfy the succession of whole process, re-authorization, validation secret keys and authentication, which coincide with the results of other existing works. We also found that the integrity of messages can not be guaranteed due to auth-confirm message may encounter man-in-the-middle attacks. We proposed a solution to the problem and verified its validity.

REFERENCES

[1] Clavel, M., et al.: All about Maude, LNCS, vol. 4350. Springer (2007)
[2] Clavel, M., Durán, F., Eker, S., et al.: Maude: specification and programming in rewriting logic. Theor. Comput. Sci. 285(2), 187–243 (2002)
[3] Eker, S., Meseguer, J., Sridharanarayanan, A.: The Maude LTL model checker. In: 4th WRLA, ENTCS 71. pp. 162–187. Elsevier (2002)
[4] IEEE, B.E.: Ieee standard for local and metropolitan area networks part 16: Air interface for broadband wireless access systems amendment 3: Advanced air interface pp. 1–1112 (2011)
[5] Kahya, N., Ghoualmi, N., Lafourcade, P.: Formal analysis of PKM using scyther tool. In: International Conference on Information Technology and E-Services. pp. 1–6 (2012)
[6] Raju, K.V.K.: Formal Verification of IEEE802.16m PKMv3 Protocol Using CasperFDR. In: Information and Communication Technologies - International Conference. pp. 590–595 (2010)
[7] Sikkens, B.: Security issues and proposed solutions concerning authentication and authorization for WiMAX (IEEE 802.16e). In: Proc. of 8th Conference on IT Enschede University of Twente (2008)
[8] Xu, S., Huang, C.T.: Attacks on PKM Protocols of IEEE 802.16 and Its Later Versions. In: International Symposium on Wireless Communication Systems. pp. 185–189 (2006)
[9] Xu, S., Huang, C.T., Matthews, M.M.: Modeling and analysis of IEEE 802.16 PKM Protocols using CasperFDR. In: IEEE International Symposium on Wireless Communication Systems. pp. 653–657 (2008)
[10] Zhu, X., Xu, Y., Guo, J., Wu, X.: Formal Verification of PKMv3 Protocol Using DT-Spin. In: International Symposium on Theoretical Aspects of Software Engineering. pp. 71–78 (2015)

A Formal Design Model of Cloud Services

Meng Sun* and Guirong Fu†

*LMAM & DI, School of Mathematical Sciences, Peking University, Beijing, China
sunmeng@math.pku.edu.cn
†Yuanpei College, Peking University, Beijing, China
fgr079@126.com

***Abstract*—To support rigorous development of cloud applications, a formal model for understanding and reasoning about cloud services is needed. Unifying Theories of Programming (UTP) provide a formal semantic foundation for various expressive programming and specification languages. A key concept in UTP is design: the familiar pre / post-condition pair that describes a contract. In this paper we use UTP to provide a formal model for cloud computing, whereby cloud services are interpreted as designs in UTP. Refinement and equivalence relations between cloud services can be naturally established by implication between predicates. A family of composition operators that can be used to put different cloud services together to construct more complex services and applications are defined based on the design model for cloud services. On the other hand, dynamic reconfiguration of cloud applications can be dealt with in the context of the design model as well, by applying the reconfiguration rules on the design models for the corresponding applications.**

Keywords: Cloud service, design, composition, refinement, dynamic reconfiguration

I. Introduction

Cloud computing has been coined as an umbrella term to describe a family of sophisticated computing services and gained a significant amount of attention in the past decades. It denotes a model on which a computing infrastructure is viewed as a "cloud", from which businesses and individuals access applications from different locations. According to [3], Cloud is defined as a parallel and distributed computing system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements (SLAs). Instead of running or storing applications locally, one can host their application in the cloud and access it from any location using a client such as a web browser.

To support rigorous development of cloud applications, we need to investigate the formal foundations for understanding and reasoning about clouds. Although researches on cloud computing are mainly focused on technical problems such as resource allocation [9], resource sharing [12], resource management [21], policy optimization [16] and task scheduling [22], there are some attempts to the formalization of fundamental notions in cloud computing. An abstract formal model of cloud workflows was proposed in [8] using the Z notation. In [7], the hierarchical colored Petri Net model was adopted to specify the security mechanism in cloud computing. The Petri Net model is also used in [4] to build the fault tolerant model of cloud computing, and as the basis for a dynamic fault tolerant strategy in cloud computing. The agent paradigm was adopted in [19] to manage cloud resources and support cloud service discovery, negotiation and composition. A Bigraph model was proposed in [2], [18] to formally specify cloud services and customers and their interaction schemes. A formal model for aspects of performance, resource consumption, and deployment on the cloud is developed using the abstract behavioral specification language ABS in [5]. In [1], the model checker UPPAAL is used to synthesize an optimal infinite scheduler for a given specification of Mobile Cloud Computing systems.

The problem that we address in this paper is to develop a formal model for cloud services under the UTP (Unifying Theories of Programming) semantic framework. UTP was proposed by Hoare and He in [10], and aims to formalize the similar features of different languages in a similar style. UTP has been proved to be appropriate for formal semantics of various programming languages and specification languages like Circus [15], TCOZ [17], rCOS [11] and Reo [20]. We believe it is also well suited for developing a proper formal foundation of cloud computing. Furthermore, interpreting cloud services as UTP designs makes it easier to guarantee the consistency of cloud service specifications and the corresponding implementations in different concrete languages and platforms whose semantics can also be given in UTP, which is very important for Cloud computing.

Clouds provide applications / storages / services that can be used by local clients from different locations in the real world, and clients can link to the cloud via different instruments. Since our observation on the cloud can only be obtained by interactions between users and services provided by the cloud, for an arbitrary cloud service $\mathbf{C}$ the relevant observations usually come in pairs on its input ports and output ports respectively. When a user send a requirement to the cloud to acquire some service, the requirement will be distributed to some available resource and dealt with in the cloud. Afterwards, the result is sent out to the user and the resource is released and available again to be invoked later. Thus cloud services can be interpreted as *designs* in UTP, i.e., pairs of predicates $P \vdash Q$, where P is a predicate specifying the relationship among what happen as the inputs of the cloud service and Q is the predicate specifying the condition that should be satisfied by the outputs.

The design model for cloud services proposed in this paper provides a family of composition operators to compose different cloud services or resources to build complex applications. An advantage of representing cloud services as such designs in

DOI reference number: 10.18293/SEKE2017-055.

our model is that cloud applications can be decomposed into simpler services, and in suitable circumstances the behavior of the whole application can be captured by the composition of the predicates describing its component services. It is natural to verify cloud service properties by assume-guarantee reasoning based on the design model framework and the verification of separate services are becoming quite easy. Furthermore, dynamic reconfigurations of cloud applications can also be captured by the reconfiguration rules for transforming the design models.

The paper is organized as follows. Section II shows how cloud services are interpreted as designs in UTP based on observations on their input and output ports, and introduces the refinement and equivalence relations between cloud services. Then, in Section III we provide a family of composition operators that can be used to put different cloud services together to construct more complex services and applications. Section IV introduces dynamic reconfiguration for cloud services based on the design model. Finally, Section V summarizes the paper and comes up with some future work we are going to work on.

II. Cloud Services as Designs

Usually the computing and storage resources in the cloud are located far from the users. Users have no knowledge about its details and configuration, and can access the cloud applications regardless of their locations or what device being used. Thus the only possible way that users can know about a cloud application is via observations on the services provided by the application: A cloud service is interpreted as a relation between an initial observation on inputs to the cloud service and a subsequent observation of the behavior of the execution.

During observations it is usual to wait for some initial transient behavior to stabilize before making any further observation. To express this, two Boolean variables ok and ok' are introduced, where ok stands for a successful initialization of computation in the cloud service or communication with other services by external users or organizations, and ok' denotes the observation that the cloud service has either terminated or reached an intermediate stable state. When ok' is false, the cloud service becomes divergent.

A. Cloud Services as Designs

In this paper we use $in_{\mathbf{C}}$ and $out_{\mathbf{C}}$ to denote what happen as inputs and outputs of a cloud service $\mathbf{C}$, respectively. For every port of a cloud service $\mathbf{C}$, the corresponding observation on it is given by a timed data stream, which is defined as follows:

Definition 1: Let D be a set of data elements and $\mathbb{R}_+$ be the set of non-negative real numbers which is used to represent time moments. Let $DS = D^\omega$ be the set of data streams, that is, the set of all streams $\alpha = (\alpha(0), \alpha(1), \alpha(2), \ldots)$ over D, and $\mathbb{R}_+^\omega$ be the set of all streams $a = (a(0), a(1), a(2), \ldots)$ over $\mathbb{R}_+$. The set of time streams is defined by the following subset of $\mathbb{R}_+$:

$$TS = \{a \in \mathbb{R}_+^\omega \mid a < a'\}$$

where a' is the derivative of a defined as

$$a' = (a(1), a(2), a(3), \ldots)$$

for $a = (a(0), a(1), a(2), \ldots)$, and for two time streams a and b, $a < b \equiv \forall n \geq 0.a(n) < b(n)$. A *timed data stream* is defined as a pair $\langle \alpha, a \rangle$ consisting of a data stream $\alpha \in DS$ and a time stream $a \in TS$. We use TDS to denote the set of timed data streams.

Let $I_{\mathbf{C}}$ and $O_{\mathbf{C}}$ be the set of input and output port names of $\mathbf{C}$, then $in_{\mathbf{C}}$ and $out_{\mathbf{C}}$ are defined as the following mappings from the corresponding port sets to TDS.

$$\begin{aligned} in_{\mathbf{C}} : &\quad I_{\mathbf{C}} \to TDS \\ out_{\mathbf{C}} : &\quad O_{\mathbf{C}} \to TDS \end{aligned}$$

Definition 2: A design is a pair of predicates $P \vdash Q$, where neither predicate contains ok or ok', and P has only input variables. It has the following meaning:

$$P \vdash Q \;\equiv\; ok \wedge P \Rightarrow ok' \wedge Q$$

We use relations on timed data streams to model cloud services. Every cloud service $\mathbf{C}$ can be represented by the design $P(in_{\mathbf{C}}) \vdash Q(in_{\mathbf{C}}, out_{\mathbf{C}})$. In the following we write such a design for cloud service $\mathbf{C}$ as:

$$\begin{aligned} &\mathbf{C}(in : in_{\mathbf{C}}; out : out_{\mathbf{C}}) \\ \mathbf{pre} &: P(in_{\mathbf{C}}) \\ \mathbf{post} &: Q(in_{\mathbf{C}}, out_{\mathbf{C}}) \end{aligned}$$

where $\mathbf{C}$ is the name of the cloud service, $P(in_{\mathbf{C}})$ is the condition that should be satisfied by inputs $in_{\mathbf{C}}$ of the cloud service, and $Q(in_{\mathbf{C}}, out_{\mathbf{C}})$ is the condition that should be satisfied by outputs $out_{\mathbf{C}}$ of $\mathbf{C}$.

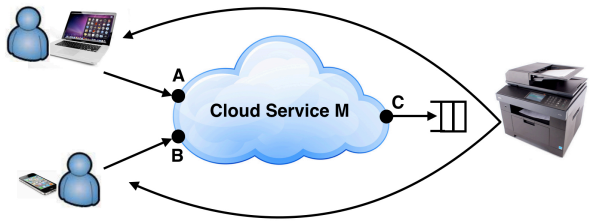


Fig. 1. Remote Printing Service

Example 1: Consider a simple example where a remote printer offers its printing service to two clients, which compete for the use of this shared resource. Each client can send out multiple printing requests to the printer and the requests from different clients are placed in a queue to be processed by the printer in a first-come first-served manner. After a file is printed out it can be collected by the client later. In order to keep the example simple to expose without considering non-deterministic choice, we assume that requests from different clients never arrive simultaneously.

The cloud service $\mathbf{M}$ in Figure 1 receives requests from different clients at ports A and B, and delivers a sequence of requests through port C to a queue on the printer side. The specification of such a service is given as follows:

$$\begin{aligned} &\mathbf{M}(in : (A \mapsto \langle \alpha, a \rangle, B \mapsto \langle \beta, b \rangle); out : C \mapsto \langle \gamma, c \rangle) \\ \mathbf{pre} &: \mathcal{D}(\langle \alpha, a \rangle) \wedge \mathcal{D}(\langle \beta, b \rangle) \wedge \forall i, j \geq 0.a(i) \neq b(j) \\ \mathbf{post} &: \mathcal{D}(\langle \gamma, c \rangle) \wedge M(\langle \alpha, a \rangle, \langle \beta, b \rangle, \langle \gamma, c \rangle) \end{aligned}$$

where $\mathcal{D}(\langle\alpha, a\rangle)$ is a predicate to judge whether $\langle\alpha, a\rangle$ is a well-defined TDS satisfying the requirements on the corresponding port, and $M(\langle\alpha, a\rangle, \langle\beta, b\rangle, \langle\gamma, c\rangle)$ is a predicate that captures the behavior of merging two timed data streams $\langle\alpha, a\rangle$ and $\langle\beta, b\rangle$ into $\langle\gamma, c\rangle$, and is defined as follows:

$$
\begin{aligned}
&M(\langle\alpha, a\rangle, \langle\beta, b\rangle, \langle\gamma, c\rangle) \\
&= \begin{cases}
\langle\gamma, c\rangle = \langle\alpha, a\rangle & \text{if } |\langle\beta, b\rangle| = 0 \\
\langle\gamma, c\rangle = \langle\beta, b\rangle & \text{if } |\langle\alpha, a\rangle| = 0 \\
\begin{cases}
\gamma(0) = \alpha(0) \wedge c(0) = a(0) \wedge \\
M(\langle\alpha', a'\rangle, \langle\beta, b\rangle, \langle\gamma', c'\rangle) \text{ if } a(0) < b(0) \\
\gamma(0) = \beta(0) \wedge c(0) = b(0) \wedge \\
M(\langle\alpha, a\rangle, \langle\beta', b'\rangle, \langle\gamma', c'\rangle) \text{ if } b(0) < a(0)
\end{cases} \\
\text{otherwise}
\end{cases}
\end{aligned}
$$

B. Refinement and Equivalence of Cloud Services

The notion of refinement has been widely used in different kinds of system descriptions. For example, in data refinement [6], the concrete model is required to have enough redundancy to completely represent the abstract model. Such a relationship is guaranteed by a surjective map from the concrete model to the abstract one. Implication of predicates establishes a proper refinement order over cloud services. Thus, more concrete implementations imply more abstract specifications. For two cloud services $\mathbf{C}_1$ and $\mathbf{C}_2$, if $in_{\mathbf{C}_1} = in_{\mathbf{C}_2}$ and $out_{\mathbf{C}_1} = out_{\mathbf{C}_2}$, then

$$\mathbf{C}_1 \sqsubseteq \mathbf{C}_2 \ =_{df} \ (P_1 \Rightarrow P_2) \wedge (P_1 \wedge Q_2 \Rightarrow Q_1) \qquad (1)$$

In other words, preconditions on inputs of cloud services are weakened under refinement, and postconditions on outputs of cloud services are strengthened. Taking equation (1) into consideration, $\mathbf{C}_2$ is stronger than $\mathbf{C}_1$ because it has a weaker assumption P_2, and so it can be used in more contexts. Furthermore, in all circumstances where $\mathbf{C}_1$ can be applied, $\mathbf{C}_2$ has a stronger commitment, so its behavior can be more precisely predicted and controlled comparing with $\mathbf{C}_1$.

Equivalence of cloud services is defined in the normal way by mutual refinement:

$$\mathbf{C}_1 \equiv \mathbf{C}_2 \quad \textbf{iff} \quad \mathbf{C}_1 \sqsubseteq \mathbf{C}_2 \wedge \mathbf{C}_2 \sqsubseteq \mathbf{C}_1$$

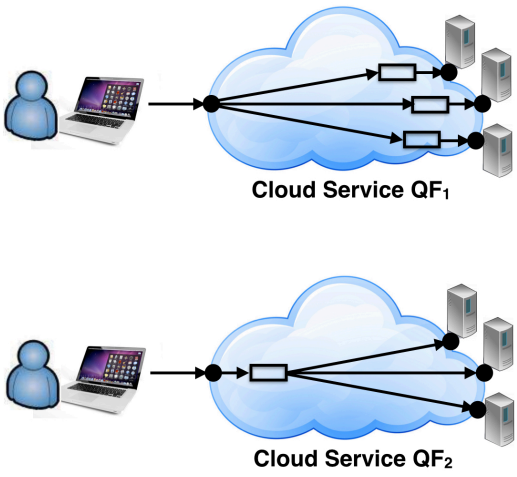


Fig. 2. Refinement of Query Flight Service

Example 2: Consider a cloud application where clients can check flight information and order tickets. If the client plan to make a trip between two places and make a query for the flight information, because there can be many flights between these two places provided by different companies, and the availability and price for each flight may change at any time, the client hope to collect the information for all available flights at the latest time. We first take the service $\mathbf{QF_1}$ into consideration. It has a buffer on the server side for each flight company, accepts query from the client and put a copy of the query into every buffer. We have its specification by design model as follows:

$$
\begin{aligned}
&\mathbf{QF}_1(in : A \mapsto \langle\alpha, a\rangle; out : (B_1 \mapsto \langle\beta_1, b_1\rangle, \\
&\qquad B_2 \mapsto \langle\beta_2, b_2\rangle, B_3 \mapsto \langle\beta_3, b_3\rangle)) \\
&\textbf{pre} : \mathcal{D}(\langle\alpha, a\rangle) \\
&\textbf{post} : \bigwedge_{1 \le i \le 3} (\mathcal{D}(\langle\beta_i, b_i\rangle) \wedge \beta_i = \alpha \wedge a < b_i < a')
\end{aligned}
$$

Then we can consider another cloud service $\mathbf{QF_2}$ in Figure 2, whose buffer's location is on the client's side, which is different from $\mathbf{QF_1}$. It can accept the query from the client as well, puts it into the buffer and sends a copy of the query to each flight company simultaneously. The trick here is the query waits in the buffer and be delivered to each server end simultaneously. The design model description of it is as follows:

$$
\begin{aligned}
&\mathbf{QF}_2(in : A \mapsto \langle\alpha, a\rangle; out : (B_1 \mapsto \langle\beta_1, b_1\rangle, \\
&\qquad B_2 \mapsto \langle\beta_2, b_2\rangle, B_3 \mapsto \langle\beta_3, b_3\rangle)) \\
&\textbf{pre} : \mathcal{D}(\langle\alpha, a\rangle) \\
&\textbf{post} : \bigwedge_{1 \le i \le 3} (\mathcal{D}(\langle\beta_i, b_i\rangle) \wedge \beta_i = \alpha) \wedge a < b_1 = b_2 = b_3 < a')
\end{aligned}
$$

From the two design models we can easily derive that $\mathbf{QF_1} \sqsubseteq \mathbf{QF_2}$ since their preconditions are equal while the postcondition for $\mathbf{QF_2}$ is stronger.

III. Composition of Cloud Services

Different cloud services can be composed together to build more complex services / applications. Since cloud services are interpreted as designs, their composition can be naturally modeled by composition on designs, which leads to a new design capturing the behavior of the composed cloud service / application. In this section, we introduce a family of composition patterns for two cloud services $\mathbf{C}_i (i = 1, 2)$:

$$
\begin{aligned}
&\mathbf{C}_i(in : in_{\mathbf{C}_i}; out : out_{\mathbf{C}_i}) \\
&\textbf{pre} : P_i(in_{\mathbf{C}_i}) \\
&\textbf{post} : Q_i(in_{\mathbf{C}_i}, out_{\mathbf{C}_i})
\end{aligned}
$$

In the following, we use P_i and Q_i instead of $P_i(in_{\mathbf{C}_i})$ and $Q_i(in_{\mathbf{C}_i}, out_{\mathbf{C}_i})$ for simplicity of expression when it is clear from the context.

A. Sequential composition

Suppose one output port O of $\mathbf{C}_1$ and one input port I of $\mathbf{C}_2$ can be joined together and the timed data stream that happen on O thus can be taken as the input on I for $\mathbf{C}_2$. After joining these two ports, what happened on O (and I)

will be hidden from outside, which can be specified by using existential quantification on the corresponding predicates. Let the output on O in $\mathbf{C}_1$ and the input on I in $\mathbf{C}_2$ be $O \mapsto \langle \delta_1, d_1 \rangle \in out_{\mathbf{C}_1}$ and $I \mapsto \langle \delta_2, d_2 \rangle \in in_{\mathbf{C}_2}$, respectively. Then the results cloud service by sequentially composing $\mathbf{C}_1$ and $\mathbf{C}_2$ is:

$$\begin{aligned} &\mathbf{C}_1;_{O \to I} \mathbf{C}_2(in : \bigcup_{i=1,2} in_{\mathbf{C}_i} \setminus \{I \mapsto \langle \delta_2, d_2 \rangle\}; \\ &\qquad out : \bigcup_{i=1,2} out_{\mathbf{C}_i} \setminus \{O \mapsto \langle \delta_1, d_1 \rangle\}) \\ &\mathbf{pre} : P_1 \wedge \neg(Q_{1\,\langle \delta_1, d_1 \rangle};_{\langle \delta_2, d_2 \rangle} \neg P_2) \\ &\mathbf{post} : Q_{1\,\langle \delta_1, d_1 \rangle};_{\langle \delta_2, d_2 \rangle} Q_2 \end{aligned}$$

where the sequential composition of predicates is defined as follows:

$$\begin{aligned} &P_{\langle \delta_1, d_1 \rangle};_{\langle \delta_2, d_2 \rangle} Q \\ &\equiv \exists \langle \delta, d \rangle . P[\langle \delta, d \rangle / \langle \delta_1, d_1 \rangle] \wedge Q[\langle \delta, d \rangle / \langle \delta_2, d_2 \rangle] \end{aligned}$$

For $in_{\mathbf{C}_i} : I_{\mathbf{C}_i} \to TDS, i = 1, \ldots, k$,

$$\bigcup_{i=1,\ldots,k} in_{\mathbf{C}_i} : \bigcup_{i=1,\ldots,k} I_{\mathbf{C}_i} \to TDS$$

is defined as:

$$\bigcup_{i=1,\ldots,k} in_{\mathbf{C}_i}(K) = in_{\mathbf{C}_j}(K) \text{ if } K \in I_{\mathbf{C}_j}.$$

And for an arbitrary port K,

$$\bigcup_{i=1,\ldots,k} in_{\mathbf{C}_i} \setminus \{K \mapsto \langle \delta, d \rangle\} = (\bigcup_{i=1,\ldots,k} in_{\mathbf{C}_i})_{\bigcup_{i=1,\ldots,k} I_{\mathbf{C}_i} \setminus K}$$

Definitions for union and subtraction on outputs are similar.

For a predicate P and a variable v in P, $P[u/v]$ is the predicate obtained by replacing all the occurrence of v in P by u. Note that when two cloud services $\mathbf{C}_1$ and $\mathbf{C}_2$ are sequentially composed, we can certainly join more than one pair of ports together and the definition of the resulting service is similar, but it is not necessary to join all the output ports of $\mathbf{C}_1$ to all the input ports of $\mathbf{C}_2$. Some ports in the services can be left as the input / output ports for the resulting service. The definition for the general situation is similar and can be easily obtained.

B. External, internal and conditional choices

Cloud services can be aggregated in a number of different ways, besides the sequential composition. In the following we consider a few such combinators. A typical composition pattern being widely used is *external choice*. For the two cloud services $\mathbf{C}_1$ and $\mathbf{C}_2$, when they are put together and interacting with the environment, clients from the environment are allowed to choose either to input on the input ports of $\mathbf{C}_1$, or on input ports of $\mathbf{C}_2$, which will trigger the corresponding cloud service $\mathbf{C}_1$ or $\mathbf{C}_2$, respectively, and produce the associated output on the corresponding output ports. Formally, the results cloud service as an external choice of $\mathbf{C}_1$ and $\mathbf{C}_2$ is defined as:

$$\begin{aligned} &\mathbf{C}_1 \,\square\, \mathbf{C}_2(in : \bigcup_{i=1,2} in_{\mathbf{C}_i}; out : \bigcup_{i=1,2} out_{\mathbf{C}_i}) \\ &\mathbf{pre} : P_1 \vee P_2 \\ &\mathbf{post} : (P_1 \Rightarrow Q_1) \wedge (P_2 \Rightarrow Q_2) \end{aligned}$$

Sometimes it is possible that both cloud services might have input ports in common so that there is no clear prescription as to which route is followed when one of these common ports is chosen. In the implementation, either service can be chosen to be executed. This case is captured by the *internal choice* pattern, which is formally defined as follows:

$$\begin{aligned} &\mathbf{C}_1 \sqcap \mathbf{C}_2(in : \bigcup_{i=1,2} in_{\mathbf{C}_i}; out : \bigcup_{i=1,2} out_{\mathbf{C}_i}) \\ &\mathbf{pre} : (in_{\mathbf{C}_1} \cap in_{\mathbf{C}_2} \neq \emptyset) \wedge P_1 \wedge P_2 \\ &\mathbf{post} : Q_1 \vee Q_2 \end{aligned}$$

Besides the external and internal choices, a further form of choice, the *conditional choice* which is based on the value of a boolean expression, is also needed for combination of cloud services. This case is formally defined by the following design which means that if b is satisfied then the cloud service $\mathbf{C}_1$ is executed, and otherwise, $\mathbf{C}_2$ is executed:

$$\begin{aligned} &\mathbf{C}_1 \lhd b \rhd \mathbf{C}_2(in : \bigcup_{i=1,2} in_{\mathbf{C}_i}; out : \bigcup_{i=1,2} out_{\mathbf{C}_i}) \\ &\mathbf{pre} : P_1 \lhd b \rhd P_2 \\ &\mathbf{post} : Q_1 \lhd b \rhd Q_2 \end{aligned}$$

C. Parallel composition

After the study of the choice combinators we proceed to that of *parallel composition*. The simplest form of parallel combinator captures the case that both cloud services $\mathbf{C}_1$ and $\mathbf{C}_2$ are invoked and executed in parallel when triggered by a pair of inputs on the corresponding input ports of both $\mathbf{C}_1$ and $\mathbf{C}_2$. Therefore, to make it possible to execute the parallel combination of $\mathbf{C}_1$ and $\mathbf{C}_2$, both P_1 and P_2 should be satisfied and the execution will lead to the result that $Q_1 \wedge Q_2$.

$$\begin{aligned} &\mathbf{C}_1 \parallel \mathbf{C}_2(in : \bigcup_{i=1,2} in_{\mathbf{C}_i}; out : \bigcup_{i=1,2} out_{\mathbf{C}_i}) \\ &\mathbf{pre} : P_1 \wedge P_2 \\ &\mathbf{post} : Q_1 \wedge Q_2 \end{aligned}$$

In the parallel composition defined above, when two cloud services are put into parallel, they may evolve completely autonomously, i.e., we have no restriction on the inputs for the two services and they can arrive at any time. Sometimes we may hope to have some inputs for $\mathbf{C}_1$ and $\mathbf{C}_2$ arrive only at the same time. For simplicity, we assume that the data can only arrive at the input ports I_1 and I_2 simultaneously, where I_1 and I_2 belong to the input ports of $\mathbf{C}_1$ and $\mathbf{C}_2$ respectively. And the data arriving at all the other input ports except I_1 and I_2 are independent. Furthermore, we assume that $I_i \mapsto \langle \delta_i, d_i \rangle \in in_{\mathbf{C}_i}$ for $i = 1, 2$. Then we have

$$\begin{aligned} &\mathbf{C}_1 \;{}_{I_1}\!\parallel_{I_2} \mathbf{C}_2(in : \bigcup_{i=1,2} in_{\mathbf{C}_i}; out : \bigcup_{i=1,2} out_{\mathbf{C}_i}) \\ &\mathbf{pre} : P_1 \wedge P_2 \wedge d_1 = d_2 \\ &\mathbf{post} : Q_1 \wedge Q_2 \end{aligned}$$

In many cases, a family of cloud services may exist and behave in parallel in a pairwise fashion. To model this, the n-ary version of both parallel combinators $\parallel$ and ${}_{I_1}\!\parallel_{I_2}$ are very helpful. The definition of $\parallel$ and ${}_{I_1}\!\parallel_{I_2}$ can be easily generalized

to the case for composing multiple services and we will omit the technical details here.

A similar situation we consider is the case of merging two input ports of cloud services $\mathbf{C}_1$ and $\mathbf{C}_2$ into one port. Let $\langle \delta_i, d_i \rangle$ for $i = 1, 2$ be the timed data streams on the input port I_i in $\mathbf{C}_i$, respectively. By merging I_1 and I_2 into one port I, when the resulting service receives a request on I, it will behave in a "broadcasting" way. In other words, the request will be replicated on I and sent to both $\mathbf{C}_1$ and $\mathbf{C}_2$ to trigger their execution simultaneously. The definition of this operation is as follows:

$$\mathbf{C}_1 \|_{\{I_1, I_2\} \gg I} \mathbf{C}_2(in : (\bigcup_{i=1,2} in_{\mathbf{C}_i} \setminus \{I_i \mapsto \langle \delta_i, d_i \rangle\}) \cup \{I \mapsto \langle \delta, d \rangle\}; out : \bigcup_{i=1,2} out_{\mathbf{C}_i})$$

$$\mathbf{pre} : \bigwedge_{i=1,2} P_i[\langle \delta, d \rangle / \langle \delta_i, d_i \rangle]$$

$$\mathbf{post} : \bigwedge_{i=1,2} Q_i[\langle \delta, d \rangle / \langle \delta_i, d_i \rangle]$$

Based on the design model of cloud services, we can develop various (equivalence and refinement) laws for cloud services, especially for those composed by applying the combinators defined previously, and encode them as theorems to support a reasoning system in theorem provers like Coq or PVS. However, instead of proceeding further with such laws, we shall discuss about dynamic reconfiguration of cloud applications in the following section, which is a rather important topic for Cloud computing.

IV. Dynamic Reconfiguration

Dynamic reconfiguration is necessary in Cloud computing. For example, when we consider a cloud application that uses cloud services provided by different providers, where the services are usually not under our control, and thus cannot be reset in general. During the execution of such an application, it is not surprising that some service provider becomes unavailable or the QoS properties of some service becomes much worse, which might be unacceptable for the clients of the application. In such cases, a dynamic reconfiguration is certainly necessary and can be either very simple like switching to an alternative cloud service, or very complex like modifying the whole application architecture.

In our approach, we model dynamic reconfigurations of a cloud application using a family of reconfiguration rules on the corresponding design model. A reconfiguration rule captures a possible pattern that should be matched by the reconfiguration, and specifies how the application should be changed during the reconfiguration. It is obvious that we cannot provide a complete set of rules for all possible reconfiguration situations here. Instead, we only consider the basic reconfiguration situation for adding new services during the execution, which is rather simple but quite common in Cloud computing. Details about the operation are given in the following of this section.

Suppose we have a cloud application $\mathbf{S}$, and $\mathbf{C}_1$ is a service being used in this application. Another service $\mathbf{C}_2$ which behaves like $\mathbf{C}_1$ might be offered as another option for use when $\mathbf{C}_1$ is invoked during the execution of the application, while $\mathbf{C}_1$ is still available there. When the request to execute service $\mathbf{C}_1$ is coming, either $\mathbf{C}_1$ or $\mathbf{C}_2$ will be invoked and return the corresponding results. Such a situation is captured by the reconfiguration rule for adding service as another option to choose:

Rule 1: $\mathbf{S}[\![\mathbf{C}_1]\!] \rightsquigarrow \mathbf{S}[\![\mathbf{C}_1 \between \mathbf{C}_2]\!]$.

This rule means that the service $\mathbf{C}_1$ is replaced by $\mathbf{C}_1 \between \mathbf{C}_2$ in the application $\mathbf{S}$, while all the other parts in $\mathbf{S}$ are kept unchanged in the reconfiguration. Let $\mathbf{C}_i(in : (I_i \mapsto \langle \alpha_i, a_i \rangle); out : (O_i \mapsto \langle \beta_i, b_i \rangle))$, $i = 1, 2$, be two cloud services, then we have *:

$$\mathbf{C}_1 \between \mathbf{C}_2 = \mathbf{Router};_{\{K_i \to I_i\}} (\mathbf{C}_1 \square \mathbf{C}_2);_{\{O_i \to J_i\}} \mathbf{Merger}$$

where $\mathbf{Router}$ and $\mathbf{Merger}$ are defined as follows:

$$\mathbf{Router}(in : (I_1 \mapsto \langle \alpha, a \rangle); out : (K_1 \mapsto \langle \alpha_1, a_1 \rangle, K_2 \mapsto \langle \alpha_2, a_2 \rangle))$$

$$\mathbf{pre} : \mathcal{D}(\langle \alpha, a \rangle)$$

$$\mathbf{post} : (\bigwedge_{i=1,2} \mathcal{D}(\langle \alpha_i, a_i \rangle)) \wedge M(\langle \alpha_1, a_1 \rangle, \langle \alpha_2, a_2 \rangle, \langle \alpha, a \rangle)$$

and

$$\mathbf{Merger}(in : (J_1 \mapsto \langle \beta_1, b_1 \rangle, J_2 \mapsto \langle \beta_2, b_2 \rangle); out : (O_1 \mapsto \langle \beta, b \rangle))$$

$$\mathbf{pre} : \bigwedge_{i=1,2} \mathcal{D}(\langle \beta_i, b_i \rangle)$$

$$\mathbf{post} : \mathcal{D}(\langle \beta, b \rangle) \wedge M(\langle \beta_1, b_1 \rangle, \langle \beta_2, b_2 \rangle, \langle \beta, b \rangle)$$

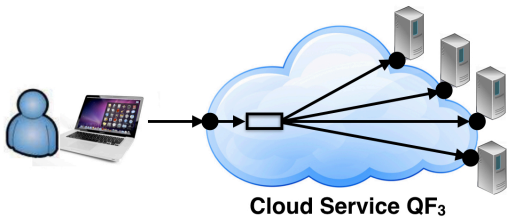


Fig. 3. Dynamic Reconfiguration of Query Flight Service

We can also add a service in parallel with another existing service. Just for a brief illustration, we can consider the flight query application depicted in Example 2. When the application is running, it is possible that some new flight company provide the flight query service for flights provided by this company, which is not available for the application at the beginning. In this case, the application should still work properly when the new service is added to the application and return the flight information for possible flights of all the companies, including the new one. Such a situation is captured by Figure 3, which specifies the result after reconfiguration on $\mathbf{QF}_2$.

*For simplicity of representation here we assume that both $\mathbf{C}_1$ and $\mathbf{C}_2$ have only one input port and one output port, while the definition can be easily generalized to cloud services with multiple input and output ports.

It is also possible that some services are becoming unavailable during the execution of a cloud application. In this case, we can have similar rules to remove a service from the application, which is either running in parallel with other services, or just as an option for choice.

In retrospect, the design model illustrated here seems promising either to capture known reconfiguration patterns, or to identify new ones, for a variety of cloud applications. A lot of work, however, remains to be done.

V. Conclusion

In this paper we define a comprehensive formal model for cloud services, their composition and dynamic reconfiguration under the UTP framework. Cloud services are interpreted as designs in UTP, which is a pair of predicates specifying the relationship between inputs and outputs. A number of combinators are defined corresponding to different ways of combining cloud services, and dynamic reconfiguration of cloud applications can be specified as rules for transforming between different designs.

Prospects for future work include the investigation of other features present in cloud computing, such as service discovery, coordination and negotiation, as well as the planning of significative case studies to assess empirically the merits of the approach proposed here by linking the model to concrete implementations. We are also interested in simulation of application behavior. In addition, another natural follow up of this paper concerns reasoning about cloud applications in theorem provers like PVS or Coq, extending our previous work on formal reasoning about component connectors (e.g., in [13]) to take into account behavior of services. Developing rules for composition and dynamic reconfiguration of cloud services is in our plan as well. In particular, we would like to tackle the consistency problem among different applications / services, and to explore the relationship between reconfiguration and other kinds of model transformation, such as architectural refinement [14].

Acknowledgements

The work was partially supported by the National Natural Science Foundation of China under grant no. 61532019, 61202069 and 61272160.

References

[1] L. Aceto, K. G. Larsen, A. Morichetta, and F. Tiezzi. A Cost/Reward Method for Optimal Infinite Scheduling in Mobile Cloud Computing. In *Proceedings of FACS 2015*, volume 9539 of *LNCS*, pages 66–85. Springer, 2016.

[2] Z. Benzadri, F. Belala, and C. Bouanaka. Towards a Formal Model for Cloud Computing. In *ICSOC 2013 Workshops*, volume 8377 of *LNCS*, pages 381–393. Springer, 2014.

[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25:599–616, 2009.

[4] L. Chen, G. Fan, and Y. Liu. Modeling and analyzing cost-aware fault tolerant strategy for cloud application. In *Proceedings of SEKE 2016*, pages 439–442. KSI Research Inc. and Knowledge Systems Institute Graduate School, 2016.

[5] F. S. de Boer, R. Hähnle, E. B. Johnsen, R. Schlatte, and P. Y. H. Wong. Formal Modeling of Resource Management for Cloud Architectures: An Industrial Case Study. In *Proceedings of ESOCC 2012*, volume 7592 of *LNCS*, pages 91–106. Springer-Verlag, 2012.

[6] W.-P. de Roever and K. Engelhardt. *Data Refinement: Model Oriented Proof Methods and their Comparison*. Cambridge University Press, 1998.

[7] D. Fitch and H. Xu. A raid-based secure and fault-tolerant model for cloud information storage. *International Journal of Software Engineering and Knowledge Engineering*, 23(05):627–654, 2013.

[8] L. Freitas and P. Watson. Formalizing workflows partitioning over federated clouds: multi-level security and costs. *International Journal of Computer Mathematics*, 91(5):881–906, 2014.

[9] M. Graiet, A. Mammar, S. Boubaker, and W. Gaaloul. Towards Correct Cloud Resource Allocation in Business Processes. *IEEE Transactions on Services Computing*, 10(1):23–36, 2017.

[10] C. A. R. Hoare and J. He. *Unifying Theories of Programming*. Prentice Hall International, 1998.

[11] H. Jifeng, X. Li, and Z. Liu. rCOS: a Refinement Calculus of Object Systems. *Theoretical Computer Science*, 365(1-2):109–142, 2006.

[12] A. Jin, W. Song, P. Wang, D. Niyato, and P. Ju. Auction Mechanisms Toward Efficient Resource Sharing for Cloudlets in Mobile Cloud Computing. *IEEE Transactions on Services Computing*, 9(6):895–909, 2016.

[13] Y. Li and M. Sun. Modeling and Verification of Component Connectors in Coq. *Science of Computer Programming*, 113(3):285–301, 2015.

[14] S. Meng, L. S. Barbosa, and Z. Naixiao. On Refinement of Software Architectures. In *Proceedings of ICTAC'05*, volume 3722 of *LNCS*, pages 469–484. Springer, 2005.

[15] M. Oliveira, A. Cavalcanti, and J. Woodcock. A Denotational Semantics for Circus. *Electronic Notes in Theoretical Computer Science*, 187:107–123, 2007.

[16] X. Pei, H. Yu, and G. Fan. Achieving Efficient Access Control via XACML Policy in Cloud Computing. In *Proceedings of SEKE 2015*, pages 110–115. KSI Research Inc. and Knowledge Systems Institute Graduate School, 2015.

[17] S. Qin, J. S. Dong, and W. Chin. A Semantic Foundation for TCOZ in Unifying Theories of Programming. In *Proceedings of FME'03*, volume 2805 of *LNCS*, pages 321–340. Springer, 2003.

[18] H. Sahli, C. Bouanaka, and A. T. E. Dib. Towards a formal model for cloud computing elasticity. In *Proceedings of 2014 IEEE 23rd International WETICE Conference*, pages 359–364. IEEE Computer Society, 2014.

[19] K. M. Sim. Agent-based cloud computing. *IEEE Transactions on Services Computing*, 5(4):564–577, 2012.

[20] M. Sun, F. Arbab, B. K. Aichernig, L. Astefanoaei, F. S. de Boer, and J. J. M. M. Rutten. Connectors as designs: Modeling, refinement and test case generation. *Science of Computer Programming*, 77(7-8):799–822, 2012.

[21] D. Weerasiri, M. C. Barukh, B. Benatallah, and J. Cao. A Model-Driven Framework for Interoperable Cloud Resources Management. In *Proceedings of ICSOC 2016*, volume 9936 of *LNCS*, pages 186–201. Springer, 2016.

[22] P. Zhang, C. Lin, X. Ma, F. Ren, and W. Li. Monitoring-Based Task Scheduling in Large-Scale SaaS Cloud. In *Proceedings of ICSOC 2016*, volume 9936 of *LNCS*, pages 140–156. Springer, 2016.

Exploring the Influence of Feature Selection Techniques on Bug Report Prioritization

Yabin Wang, Tieke He, Weiqiang Zhang, Chunrong Fang, Bin Luo*
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
*luobin@nju.edu.cn

Abstract

To improve software quality, developers often open a bug repository and allow users to find bugs, describe bugs in the form of bug reports and submit bug reports to the repository. Based on the description, testers assign a priority to each bug report. In the beginning the process of priority assignment is performed manually. With the increasing amount of bug reports, researchers introduced classification methods to assign priorities automatically with all the features considered. In this paper feature selection methods are introduced to improve the effect of bug report prioritization using classification models. The experimental results show that feature selection based on Information Gain and Pearson Correlation can improve the precision and recall for bug report prioritization on two models, i.e., SVM and Naive Bayes.

Keywords: Feature selection, Bug report prioritization, SVM, Naive Bayes

1 Introduction

Bug repositories of open software projects are often accessible to the public. Both users and developers can submit problems and suggestions on how to improve the software to the repository in the form of bug reports. A bug report is helpful for debugging in many ways. It creates a communication between bug finders and bug fixers and enables developers discuss how to fix a bug. The openness of the bug repositories makes more people being able to help to find more bugs for the developers, such that the quality of the software can be greatly improved. In order to use bug repository more effectively, bug reports should be managed. Reported bugs should be analyzed to determine whether they are valid or not, correct or not, and unique or not. This is called a bug triage process [1].

Manually validating large number of bug reports can be time-consuming and tedious. Not all the bug reports are regarded as important on the side of developers. Each bug report should be assigned a priority to indicate its importance, and this process is called bug report prioritization. Some bug reports are about meaningful bugs, while other bug reports maybe just suggestions of adding software functions. Bug reports that report meaningful bugs should have a higher priority. If the developers analyze the bug reports one by one, some important bug reports may be postponed.

The amount of bug reports are often very large. In the Wordpress project we studied, there are more than 20 thousand reports. The large amount of bug reports makes it very time consuming to manually assign priorities to bug reports. One way to solve the problem is to ask bug reporters assign the priority. But lacking the full knowledge of the whole project, bug reporters could not correctly assign priorities.

To correctly assign priority to the reports, a bug triager needs to analyze the bug description in the bug report to get the information about which component the bug may belong to, what type of the bug it is, its severity and whether it shall be solved immediately or can be postponed. According to these information, priorities are assigned to bug reports.

Currently, there have been many studies focusing on automatically prioritize bug reports. For example, DRONE [2] analyzed the textual description, author and product of bug reports to assign priorities. Kremenet et al. [3] checked the success and failure of a bug report to prioritize reports. Lamkanfi [4] used various classification algorithms such as Support Vector Machine (SVM) and Naive Bayes to assign priorities for bug reports. In their study textual descriptions were transformed to feature vectors, which were the inputs of classification models. However, the study of [1] tells that not all the features are important for bug report prioritization. Their results motivated us to adopt feature selection methods on bug report prioritization.

In this paper, the textual description of a bug report are first transformed into a feature vector, and then we use feature selection to build a subset of features that can represent the whole set. This subset can give us enough information for classification to prioritize bug reports. The subset could obtain better classification results. In this paper, 7 most popular feature selection approaches including *CfsSubset* (CFS), *Correlation* (CO), *GainRatio* (GR), *InfoGain* (IG), *OneR* (OR), *ReliefF* (RF) and *SymmetricalUncert* (SU) [5, 6] are considered. These 7 techniques cover two main feature selection techniques the wrapper methods and filter methods. The wrapper methods search the feature space and evaluate feature set to find the optimal feature subset. The filter methods evaluate single feature, score each feature and rank features according to the scores. We conducted ex-

10.18293/SEKE2017-073

periments to compare the feature selection techniques and studied the effects of the number of features selected, and we found that GR, IG and CO were more suitable for bug report prioritization.

In our experiment, we evaluated the effects of some popular feature selection techniques on bug report prioritization. Two large open source projects Trac [7] and Wordpress [8] were used.

The main contributions of this paper are summarized as follows:

- We are the first to introduce feature selection methods to find the features that are most relevant to priorities. Previous researches used all the features to classify bug reports but failed to study the important features that are related with priorities. Feature selection can find out the important features.
- We compare different feature selection techniques in bug report prioritization. The results show that Information Gain and Pearson Correlation based approaches achieve the best performance.
- We study the influence of the number of features on the feature selection techniques and find that one third to half of the features are optimal for feature selection.
- We conducted experiments on two large open projects with large amount of bug reports to validate our proposals.

The rest of the paper is organized as follows. Section 2 presents the related work. The introduction of classification is described in Section 3. The framework along with the classification model used is presented in section 4. The experiment setup, evaluation and experiment results are presented in section 5, 6 and 7. Finally we summarize the threats to validity and conclude our work in section 8 and 9.

2 Related Work

Menzies and Marcus were the first researchers that studied the bug report prioritization [9]. They analyzed the severity labels text and information of bug reports of NASA and generated 5 severity labels. They first extracted word tokens from text information of bug reports and then preprocessed the text to remove stop words and performed stemming. Then the words were transformed into feature vectors and then fed into a classification model.

Lamkanfi et al. [10] extended the work of Menzies and Marcus and studied various classification algorithms. They found that SVM and Naive Bayse were two most effective classification models. This motivated us to use these two models in our research. Khomh et al. [11] studied crash reports prioritization based on the frequency of the crashes. In this paper all types of bug reports were studied instead of only focusing on crash reports. All of existing approaches failed to emphasize the features that are relevant to priority. In this paper we introduce feature selection techniques on bug report prioritization to find the most relevant features.

3 Classification of Bug Reports

Classification of bug reports consists of two main processes. The first is training bug reports with priorities assigned correctly by triagers to build a learning model. Suppose that $R(r_1, r_2, ..., r_n)$ is a feature vector transformed from a bug report for training, $r_1, r_2, ..., r_n$ are n values of n features. Each feature is a word in the bug description, representing one dimension of information of the bug report. Each r_i represents the frequency of each word occurring in R. For each feature vector R_i, there is a special class label which represents the priority y_i of that bug report. The output of this step can be represented as a learning function.

$$y = f(R) \tag{1}$$

In function 1, $f(R)$ can be some regulations or some formulas. In the second process of testing $f(R)$ receives new bug reports with no priority labels. The output y is the predicted priority for the input bug report. In order to evaluate a classification algorithm, the bug reports with correct labels of priority will work as inputs such that precision and recall of the classification algorithm can be calculated.

4 Approach

Figure 1 shows the framework of our approach.

1 Preprocess bug reports: A commonly used text transformation method [12] is adopted to preprocess bug reports, and transform reports into feature vectors with each bug report transformed into one vector. In this phase, special symbols, brackets and punctuation are removed. Non-alphabetic words, common words and stop words are also removed, because these words are meaningless and unimportant. Stemming is then applied on the remained words to convert them into their ground meaning. The ground form of each word is seen as a feature. The frequency of occurring of each word in each bug report is generated as the value of a specific feature in a vector.

2 Split data: The feature vector set is divided into two sets, training set and testing set using five-fold cross validation [13].

3 Select features: Different feature selection techniques are applied on the training set to construct a subset of features. The outputs of feature selection are the new mapped feature vector set and the subset of features. The new mapped feature vector set only contains the information of constructed subset of features.

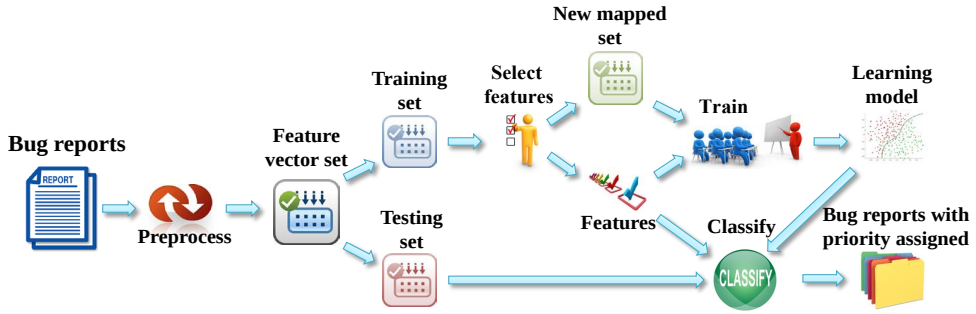


Figure 1: Framework

4 Train: Two classification algorithms, i.e., SVM and Naive Bayes are used for training the feature vectors. The output of training is the specific learning model.

5 Classify: The selected features and the learning model are used for classifying the testing set. The output is that each report is assigned a priority class.

4.1 Feature Selection for Bug Report Prioritization

There are mainly two sorts of feature selection methods, "wrapper" and "filter" methods [14]. Wrapper methods search for feature subsets of bug reports and find the subset with the highest quality [15]. Filter methods score for each feature of bug reports based on some criteria, independently. Then the features are ranked according to the score and top N features are selected [14]. In this paper 7 most popular feature selection methods *CfsSubset* (CFS), *Correlation* (CO), *GainRatio* (GR), *InfoGain* (IG), *OneR* (OR), *ReliefF* (RF) and *SymmetricalUncert* (SU) are studied. The description of the methods are as follows.

- *CfsSubset* (CFS): CFS uses Best First and Greedy Stepwise to search feature space and uses minimum description length (MDL) to measure the correlation inside a subset and correlation between the subset and the priority. The subset with the highest quality is selected [16].
- *Correlation* (CO): CO measures the Pearson Correlation between a feature and the priority to score for each feature [17].
- *InfoGain* (IG): IG scores a feature by measuring the Information Gain between a feature and the priority [18]. The concepts of Information Gain comes from information theory [19].
- *GainRatio* (GR): GR scores a feature by measuring the gain ratio with respect to the priority. GR is a measure extended from IG. The difference is that based on IG GR will further calculate the information generated by splitting the training data according to the different kinds of value of a feature [6].
- *OneR* (OR): OR evaluates a feature by measuring the classification accuracy by using OneR classifier [5].
- *ReliefF* (RF): RF evaluates a feature by repeatedly sampling a feature vector and considering the value of the given feature for the nearest vector of the same and different priority class [20].
- *SymmetricalUncert* (SU): SU evaluates a feature by measuring the symmetrical uncertainty with respect to the priority [21].

4.2 Feature Selection Based Naive Bayes

The Naive Bayes classifies bug reports by calculating the probability of a bug report assigned to a priority label using Bayes rule of conditional probability. The probability of one new bug report belonging to each priority is calculated and the priority with the highest probability is assigned to the new report.

The Naive Bays can be stated as equation 2. $P(C_i|R)$ indicates that given a report R, the probability that the priority C_i is assigned to R.

$$P(C_i|R) = (P(C_i) \times P(R|C_i))/P(R) \tag{2}$$

Naive Bayes maximizes $P(R|C_i) \times P(C_i)$ to find the priority with the highest probability for a bug report. The assumption of Naive Bayes is that a value of a feature does not depend on the value of other features. Therefore, $P(R|C_i)$ is calculated by equation 3.

$$P(R|C_i) = \prod_{j \in SF} P(r_j|C_i) \tag{3}$$

$$P(C_i|R) > P(C_j|R) \;\; j \neq i \tag{4}$$

If equation 4 is satisfied for each C_j, C_i is assigned to the new bug report. In equation 3 SF is the selected feature set. Feature selection techniques will adjust the value of $P(R|C_i)$ to affect the results of Naive Bayes by choosing an optimized SF.

4.3 Feature Selection Based SVM

SVM transforms the input feature vectors of bug reports to a higher dimension and then searches for a hyperplane with the maximum margin in the new mapped space [22]. A simple hyperplane can be defined as equation 5.

$$w \cdot x + b = 0, x = \begin{matrix} r_{11} & r_{1j} & ... & r_{1n} \\ r_{21} & r_{2j} & ... & r_{2n} \\ ... & ... & ... & \\ r_{m1} & r_{mj} & ... & r_{mn} \end{matrix}, j \in SF \quad (5)$$

In this equation, x are the feature vectors of bug reports that lie on the hyperplane, b is a scalar often referred as a bias, w is a weighting vector, $\cdot$ is the dot product, SF is the index set of selected features. Feature selection techniques adjust SF to change the hyperplane and will affect the results of SVM.

5 Experimental Setup

In this paper, we use the data set of bug reports of Trac Open Source Project and Wordpress which were popular used on the researches of bug report prioritization [23]. Bug reports of Wordpress and Trac collected between *June*, 2004 and *March*, 2013 and *August*, 2003 and *July*, 2013 were used in the research. All the bug reports have been assigned 5 classes of priorities by the triagers correctly such that precision and recall of classification algorithms can be evaluated. After preprocessing the number of features for Trac and Wordpress is $18,231$ and $14,570$. Two best classification algorithms SVM and Naive Bayes for bug report prioritization studied by Lamkanfi [4] were used on the experiments.

Project	#Bug reports	#Reporter	#Versions
Wordpress	23,848	6,013	18
Trac	10,416	4,701	11

Table 1: Data set description

6 Evaluation

In order to evaluate our approach, feature selection were performed before classification, then the precision and recall, popular evaluation criteria for bug report prioritization techniques [12], were calculated for the classification algorithms. A higher precision and recall mean that the specific feature selection techniques are more capable in finding priority relevant features. Precision is the fraction of all the predicted items that are relevant items. For a priority class C_i predicted items are all the bug reports to which C_i is assigned by the classifier. Relevant items are bug reports to which C_i should be assigned.

$$Precision = \frac{|Relevant\ items \cap Predicted\ items|}{|Predicted\ items|} \quad (6)$$

Recall is the fraction of all the relevant items that are predicted items.

$$Recall = \frac{|Relevant\ items \cap Predicted\ items|}{|Relevant\ items|} \quad (7)$$

For each priority class the precision and recall are calculated and then the average values are summarized over all priority classes. As prioritizing bug reports manually took much longer time than classification based bug report prioritization techniques did, existing researches focused on the effectiveness and did not study the efficiency and cost of classification based techniques [12].

7 Experimental Results

	Precision	Recall	Feature selection
Original	0.613	0.565	-
Rank	**0.686**	**0.679**	CO
	0.715	**0.693**	GR
	0.705	**0.654**	IG
	0.632	0.577	OR
	0.623	0.578	RF
	0.630	0.587	SU
Best First	0.654	0.54	CFS
Greedy Stepwise	0.652	0.536	CFS

Table 2: Experimental results of Naive Bayes for Trac project

	Precision	Recall	Feature selection
Original	0.572	0.592	-
Rank	**0.685**	**0.682**	CO
	0.723	**0.719**	GR
	0.618	**0.708**	IG
	0.595	0.603	OR
	0.606	0.637	RF
	0.607	0.613	SU
Best First	0.594	0.644	CFS
Greedy Stepwise	0.592	0.645	CFS

Table 3: Experimental results of Naive Bayes for Wordpress project

In order to evaluate whether feature selection can find priority relevant features, classification were performed on the original feature vector set and the set which processed by feature selection techniques. In order to get more precise results, all the experiments were conducted 30 times and the average value were recorded.

Table 2 and 3 show the experimental results of Trac project of Naive Bayes. The first column shows the search methods. Original means that original feature vector set without feature selection performed on were used for classification. Rank means that features were scored independently. Best First and Greedy Stepwise are two greedy al-

gorithms based searching methods used by CFS. The experimental results show that feature selection techniques CO, GR and IG can get higher precision and recall than the original case and the other feature selection techniques.

	Precision	Recall	Feature selection
Original	0.523	0.517	-
Rank	**0.674**	**0.689**	CO
	0.694	**0.713**	GR
	0.683	**0.668**	IG
	0.545	0.5147	OR
	0.543	0.519	RF
	0.637	0.621	SU
Best First	0.544	0.582	CFS
Greedy Stepwise	0.542	0.536	CFS

Table 4: Experimental results of SVM for Trac project

	Precision	Recall	Feature selection
Original	0.533	0.545	-
Rank	0.557	**0.672**	CO
	0.642	**0.649**	GR
	0.557	0.593	IG
	0.558	0.521	OR
	0.563	0.562	RF
	0.521	0.58	SU
Best First	0.554	0.518	CFS
Greedy Stepwise	0.552	0.526	CFS

Table 5: Experimental results of SVM for Wordpress project

Table 4 and 5 show the results of SVM for two projects. From the two tables we can get similar observations with the results of Naive Bayes. By comparing 4 tables we can see that Naive Bayes performs better than SVM.

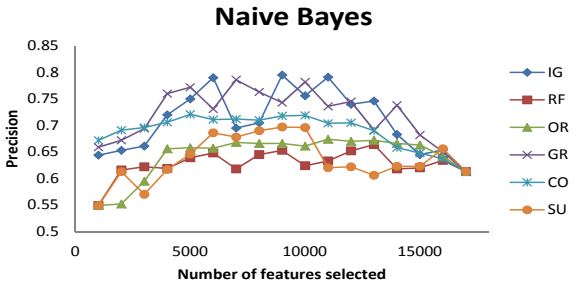


Figure 2: The influence of the number of features on precision for Trac project

Different number of features were set and the specific precision and recall were recorded for each filter method. Figure 2 shows that the precision first rises and then declines with the increase of number of features selected. The precision of GR reaches its peak when the number of features reaches about one third of the total features. This phenomenon gives us suggestions that a subset of priority relevant features are able to give enough information about the priority. We can also see that GR, IG and CO perform better than other techniques.

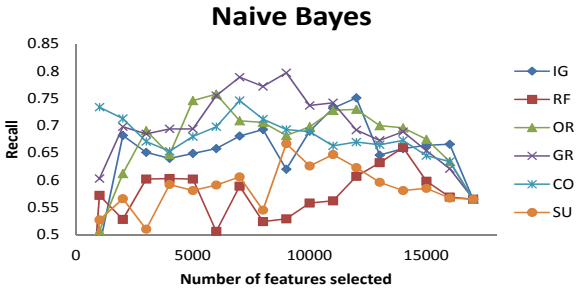


Figure 3: The influence of the number of features on recall for Trac project

In Figure 3 we can see that the peak of recall of GR occurs when the number of features reaches about half of all the features, which is later than those in Figure 2. We can also see that GR is obviously better than RF and SU.

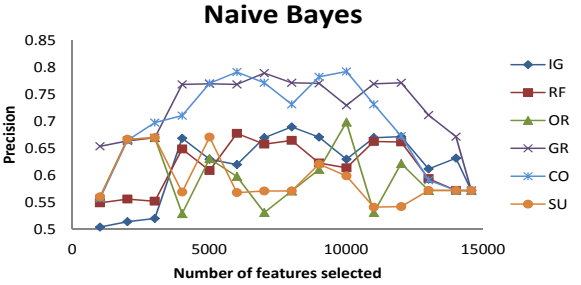


Figure 4: The effects of the number of features on precision for Wordpress project

Figure 4 and 5 show that the peak occurs when the number of features reaches about one third of all the features. Then the peak continues until about $13,000$ features selected. In these two figures CO and GR are still better than others. The performance of IG is similar with RF worse than GR. The reason is that IG dose not take the number of kinds of values of features into consideration. Features that has too many kinds of values that are not relevant with priority are also selected to make IG not as good as GR sometimes.

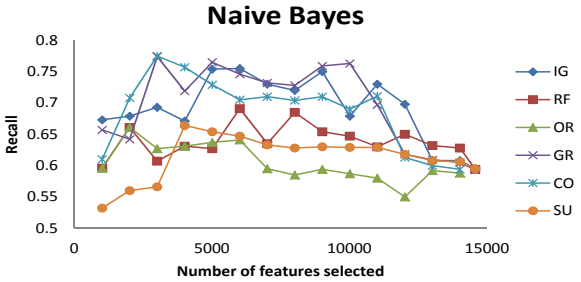


Figure 5: The influence of the number of features on recall for Wordpress project

We also used spearman correlation (SC) to measure the

correlation between the assigned priorities and the actual priorities. A higher value indicates a higher correlation. Table 6 shows the average SC of different feature selection techniques.

	CO	GR	IG	OR	RF	SU	CFS
SC	0.51	0.55	0.53	0.42	0.38	0.38	0.34

Table 6: Spearman correlation on different feature selection techniques

8 Threats to Validity

In our research the threats to validity mainly came from the experimental errors. The priority of bug reports were assigned by humans. They assigned priorities in a subjective way. Different people may have different opinions on the priorities. In the experiment we only used two projects. However, the projects were all large open source projects with large number of bug reports. This made the experiment results more universal. In the future we will study more projects and more bug reports.

9 Conclusion

In this paper feature selection methods are introduced to improve the effect of bug report prioritization using classification models. The experimental results show that feature selection can pick out relevant features and improve the effect of bug report prioritization on two models, i.e., SVM and Naive Bayes. For the feature selection techniques we studied in this paper, IG and GR based on Information Gain and CO based on Pearson Correlation obtained better performance than other feature selection techniques. We also studied the influence of the number of selected features and found that one third to half of the features were enough to get high precision and recall.

References

[1] J. Xuan, H. Jiang, Y. Hu, Z. Ren, W. Zou, Z. Luo, and X. Wu, "Towards effective bug triage with software data reduction techniques," *IEEE transactions on knowledge and data engineering*, vol. 27, no. 1, pp. 264–280, 2015.

[2] Y. Tian, D. Lo, and C. Sun, "Drone: Predicting priority of reported bugs by multi-factor analysis." in *ICSM*, 2013, pp. 200–209.

[3] T. Kremenek and D. Engler, "Z-ranking: Using statistical analysis to counter the impact of static analysis approximations," in *International Static Analysis Symposium*. Springer, 2003, pp. 295–315.

[4] A. Lamkanfi, S. Demeyer, Q. D. Soetens, and T. Verdonck, "Comparing mining algorithms for predicting the severity of a reported bug," in *Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on*. IEEE, 2011, pp. 249–258.

[5] J. Novakovic, "The impact of feature selection on the accuracy of naïve bayes classifier," in *18th Telecommunications forum TELFOR*, vol. 2, 2010, pp. 1113–1116.

[6] A. G. Karegowda, A. Manjunath, and M. Jayaram, "Comparative study of attribute selection using gain ratio and correlation based feature selection," *International Journal of Information Technology and Knowledge Management*, vol. 2, no. 2, pp. 271–277, 2010.

[7] M. D'Ambros, M. Lanza, and M. Pinzger, "" a bug's life" visualizing a bug database," in *Visualizing Software for Understanding and Analysis, 2007. VISSOFT 2007. 4th IEEE International Workshop on*. IEEE, 2007, pp. 113–120.

[8] J. Xie, M. Zhou, and A. Mockus, "Impact of triage: a study of mozilla and gnome," in *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*. IEEE, 2013, pp. 247–250.

[9] T. Menzies and A. Marcus, "Automated severity assessment of software defect reports," in *Software Maintenance, 2008. ICSM 2008. IEEE International Conference on*. IEEE, 2008, pp. 346–355.

[10] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, "Predicting the severity of a reported bug," in *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*. IEEE, 2010, pp. 1–10.

[11] F. Khomh, B. Chan, Y. Zou, and A. E. Hassan, "An entropy evaluation approach for triaging field crashes: A case study of mozilla firefox," in *Reverse Engineering (WCRE), 2011 18th Working Conference on*. IEEE, 2011, pp. 261–270.

[12] J. Kanwal and O. Maqbool, "Bug prioritization to facilitate bug report triage," *Journal of Computer Science and Technology*, vol. 27, no. 2, pp. 397–412, 2012.

[13] C.-W. Hsu, C.-C. Chang, C.-J. Lin *et al.*, "A practical guide to support vector classification," 2003.

[14] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," *Knowledge and information systems*, vol. 34, no. 3, pp. 483–519, 2013.

[15] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications*. Springer, 2008, vol. 207.

[16] M. R. Wijaya, R. Saptono, and A. Doewes, "The effect of best first and spreadsubsample on selection of a feature wrapper with naïve bayes classifier for the classification of the ratio of inpatients," *Scientific Journal of Informatics*, vol. 3, no. 2, pp. 41–50, 2016.

[17] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[18] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Transactions on neural networks*, vol. 5, no. 4, pp. 537–550, 1994.

[19] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[20] Z. Pang, D. Zhu, D. Chen, L. Li, and Y. Shao, "A computer-aided diagnosis system for dynamic contrast-enhanced mr images based on level set segmentation and relieff feature selection," *Computational and mathematical methods in medicine*, vol. 2015, 2015.

[21] S. Fong, Y. Zhuang, H. Luo, K. Liu, and G. Kim, "Finding significant factors on world ranking of e-governments by feature selection methods over kpis," in *International Conference on Soft Computing in Data Science*. Springer, 2015, pp. 65–73.

[22] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[23] Z. Xu, T. He, W. Zhang, Y. Wang, J. Liu, and Z. Chen, "Exploring the influence of time factor in bug report prioritization."

Analyzing Variability in Product Families through Canonical Feature Diagrams

Jessie Carbonnel[1], Marianne Huchard[1], and Clémentine Nebut[1]

[1]LIRMM, CNRS & University of Montpellier, France,
{jcarbonnel, huchard, nebut}@lirmm.fr

Abstract

Product line engineering aims to reduce the cost and effort to develop new related softwares, while increasing the software quality and the software scope. Variability analysis and modeling is a key issue in this approach. Several representations were proposed, including feature models (FMs) and product comparison matrices (PCMs). While PCMs are useful for presenting products in a tabular form, for their understanding and manipulation, it helps to switch to a graphical view. FMs are graphical views, but they are not canonical (i.e., several equivalent FMs can represent a same PCM) and user intervention is necessary to ensure the extraction of a meaningful FM from PCMs. In this paper, we investigate the benefits of a new structure, which captures variability in a canonical graphical representation. We outline its construction and we give insights about its shape and use when it is used as an alternative representation of wikipedia PCMs in the domain of software.

Keywords*: Product lines, Product Comparison Matrix, Feature Model, Formal Concept Analysis*

1. Introduction

Software product line engineering [8] is a software development approach which aims to produce families of similar software systems, while reducing the cost and effort, and increasing the software quality and scope. Variability analysis is a key issue in this approach that includes the organization of the main characteristics (called features) depending on the software products composing the product line. Several representations were proposed, the main one being feature models (FMs) [7]. An FM models the variability as a tree of features: a feature may introduce sub-features, and the feature is linked to a sub-feature with relation *mandatory* or *optional*, or to groups of sub-features with relation *at least one* or *exactly one* amongst the sub-features. Constraints can also be added to an FM. Other ways of modeling such as product comparison matrices (PCMs), decision models or logic formulas were also proposed, and all these representations have many variants and address specific issues. For example, FMs are appropriate representations to graphically show variability and derive product selection tools, logic formulas are relevant for automated reasoning tools and PCMs are widely used for enumerating existing representative products. While PCMs are relevant for presenting products in a tabular form, switching to a graphical point of view for their understanding and manipulation may be useful. FMs can be used in this task, but several equivalent FMs can represent a same PCM, thus their construction includes interpretation and modeling choices. Therefore, it is difficult to automate their generation from product description [1].

In this paper, we present a new structure, the Equivalence Class Feature Diagram (ECFD), which captures variability in a canonical graphical representation, and thus does not require any human interpretation to be built. The ECFD represents and structures a configuration set, and any FM depicting this configuration set conforms with the ECFD. It can be considered an interim representation between the catalog-like PCMs and the meaningful FMs. We outline the ECFD construction and we give insights about its shape and use when it is used as an alternative representation of wikipedia PCMs in the domain of software.

Next section presents the ECFD. We outline the ECFD construction in Section 3. Section 4 develops the wikipedia PCMs study. Related work is reported in Section 5. We conclude the paper with a few perspectives in Section 6.

DOI reference number: 10.18293/SEKE2017-087

2. Equivalence Class Feature Diagram (ECFD)

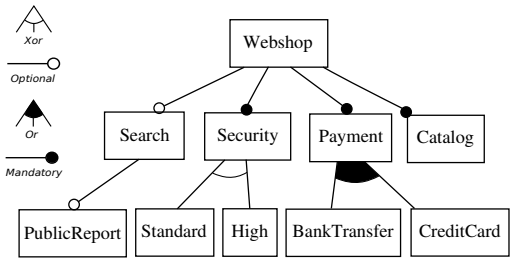
We introduce in this section Equivalence Class Feature Diagrams as an alternative way to represent variability, in a graphical and canonical way. The objective of the ECFD is to make explicit co-occurring features (features which are always present or absent together), exclusion constraints and other logical relations between features. In the following, we illustrate the ECFD with an example of e-shops configurations taken from an FM of SPLOT[1], shown in Figure 1. The configuration set is shown in Table 1 and the corresponding ECFD is given in Figure 2.

The main construct of an ECFD are boxes that represent a (maximal) set of features occurring always together. In the example of Figure 2, all the boxes are of size 1 except the one at the top, that is composed of four features. Indeed, those four features always occur together in the configuration set. There are several kinds of relations that can be added in an ECFD:

- The implications between boxes, denoted by a simple edge from a box b_1 to another box b_2, meaning that when the features of b_1 are present, the features of b_2 are also present in a configuration. In the example, we have such an implication between `PublicReport` and `Search`. Indeed in the configurations, when `PublicReport` is present, `Search` is also present.
- Or-groups can be denoted by grouped edges from several boxes b_i to another box b, to indicate that when features of b are present, there is at least the features of one box b_i. In the example, `BankTranfer` and `CreditCard` form an Or-group below the top box.
- Xor-groups are denoted like Or-groups, but with a cross on the grouped edges from several boxes b_i to another box b, to indicate that when features of b are present, there are exactly the features of one box b_i. In the example, `High` and `Standard` form a Xor-group below the top box.
- Mutex are denoted with a line linking boxes, with a cross on it. Two boxes in mutex relationship indicate that features of one box cannot appear at the same time as features of the other box. E.g., `High` and `PublicReport` are mutually exclusive.

A more detailed description of ECFD is presented in [3]. The ECFD is a canonical structure: for a given set of configurations, there is one and only one corresponding ECFD. On the contrary, the feature model is not canonical, several different feature models can represent the same set of configurations: e.g., the two feature models of Figures 1 and 3 represent the same set of configurations depicted in Table 1.

[1] http://www.splot-research.org/



$High \rightarrow !PublicReport$; $High \rightarrow CreditCard$

Figure 1. FM1 for Tang-Eshops

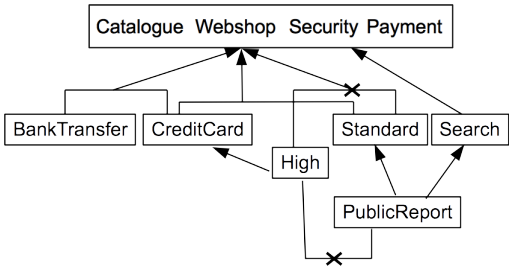
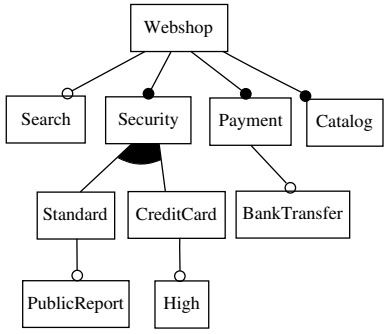


Figure 2. ECFD for Tang-Eshops

3. ECFD construction

The ECFD is built in two main steps: (1) AC-poset construction and (2) Xor-groups, Or-groups and Mutex computation. The AC-poset produces the feature equivalence classes (boxes/nodes of the ECFD) and the hierarchical structure between the nodes (Section 3.1). The Xor-groups, Or-groups and Mutex are then added to the structure (Section 3.2). This construction shares several steps with the feature model extraction proposed in [9]. However, our objectives partly diverge, since we want to preserve a canonical structure, while they aim to build one possible feature model amongst those that our canonical structure represents.



$High \rightarrow !PublicReport$; $PublicReport \rightarrow Search$
$Payment \rightarrow BankTransfer \mid CreditCard$;

Figure 3. FM2 for Tang-Eshops

3.1. AC-poset construction

The AC-poset is a structure which is defined in Formal Concept Analysis (FCA) as a sub-order of the concept lattice [5]. We briefly recall the needed definitions. Data of FCA are encoded in a *formal context* which is a triple $K = (O, A, R)$ where O and A are sets (objects and attributes, respectively) and R is a binary relation, i.e., $R \subseteq O \times A$. In our case, the objects are the configurations, and the attributes are the features (see Table 1). A *formal concept* C is a pair (E, I) composed of a maximal object set (E) and of the maximal set of attributes shared by these objects (I). $E = Extent(C) = \{o \in O | \forall a \in I, (o, a) \in R\}$ is the extent of the concept, $I = Intent(C) = \{a \in A | \forall o \in E, (o, a) \in R\}$ is the intent of the concept. The *concept lattice* is the concept set provided with a specialization order $\leqslant_s$ defined as follows: given two formal concepts $C_1 = (E_1, I_1)$ and $C_2 = (E_2, I_2)$ of K, $C_1 = (E_1, I_1) \leqslant_s C_2 = (E_2, I_2)$ if and only if $E_1 \subseteq E_2$ (and equivalently $I_2 \subseteq I_1$). C_1 is called a subconcept of C_2. C_2 is called a super-concept of C_1.

Table 1. Formal Context for Tang-Eshops

TangEshop	BankTransfer	Catalogue	CreditCard	High	Payment	PublicReport	Search	Security	Standard	Webshop
c0	×	×	×	×	×			×		×
c1		×	×		×	×	×	×	×	×
c2	×	×			×			×	×	×
c3		×	×		×		×	×	×	×
c4	×	×			×		×	×	×	×
c5	×	×	×		×		×	×	×	×
c6	×	×			×	×	×	×	×	×
c7		×	×		×			×	×	×
c8		×	×	×	×			×		×
c9	×	×	×	×	×		×	×		×
c10		×	×	×	×		×	×		×
c11	×	×	×		×			×	×	×
c12	×	×	×		×	×	×	×	×	×

Given a formal context $K = (O, A, R)$, an *attribute-concept* C is a concept such that some of the attributes of its intent are not present in any intent of any super-concept of C (these attributes are not inherited): $\exists a \in Intent(C), \forall C_{sup}, C \leqslant_s C_{sup}, a \notin Intent(C_{sup})$.

Let $\mathcal{AC}_K$ be the set of all *attribute-concepts* of a formal context K. This set of concepts provided with the specialization order $(\mathcal{AC}_K, \leqslant_s)$ is the AC-poset (for Attribute-Object-Concept poset) associated with K.

Algorithm 1 is a simple algorithm to build the Hasse diagram of the AC-poset. In this algorithm, we use complementary standard FCA notations: for any object set $S_o \subseteq O$, the set of shared attributes is $S'_o = \{a \in A | \forall o \in S_o, (o, a) \in R\}$, and for any attribute set $S_a \subseteq A$, the set of owners is $S'_a = \{o \in O | \forall a \in S_a, (o, a) \in R\}$. Figure 4 shows the AC-poset for the context of Table 1.

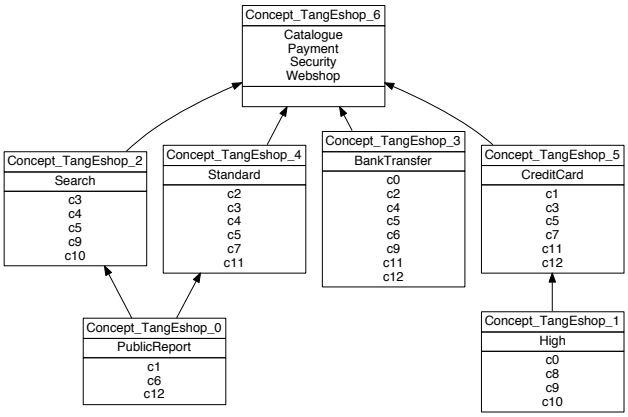


Figure 4. AC-poset for Tang-Eshops

Algorithm 1: ComputeACposet(K)

Data: K: a formal context
Result: $(AC_K, \leqslant_s)$: the AC-poset associated with K
/* attribute concepts */
1 $AC_K \leftarrow \varnothing$
2 **foreach** $a \in A$ **do**
3 $\quad AC_K \leftarrow AC_K \cup (\{a\}', \{a\}'')$ //that are the objects that share the attribute a, with all the attributes they share
4 **end**
/* specialization order */
5 Compute the transitive reduction of $\leqslant_s$ by comparing the concept extents in AC_K

While a concept lattice may have $2^{min(|O|,|A|)}$ concepts, the number of concepts in an AC-poset is bounded by $|A|$. Reference [9] details the complexity of construction of the AC-poset (concept enumeration and transitive reduction computation) which is in $\mathcal{O}(|A|^2 \times |O|^2 + |A|^3)$.

The AC-poset (as the whole concept lattice) is a canonical structure *per se*, i.e., it is unique for a given formal context (set of configurations). However, the knowledge contained in the attribute-concept extents is not easy to capture, e.g., the exclusion constraints are present, but are not emphasized. This is why we propose the ECFD, which only keeps the intents of the attribute-concepts, and adds the additional information as Xor-groups, Or-groups and Mutex in a more intuitive way.

3.2. Groups and Mutex computation

Algorithm 2 is a schematic algorithm which specifies the computation of the Xor- (alternative) and the Or-groups with some similarities with the algorithm proposed by [9]. Let us define by construction the Xor- and Or-groups which

we consider in Algorithm 2. We consider all the AC-poset antichains (sets of pairwise non comparable attribute-concepts) of growing size from 2 to $|\mathcal{AC}_K|$, which is the number of attribute-concepts (lines 2-3). For each candidate antichain A, we compute the set $Parents(A)$ containing the lowest attribute-concepts that are greater than all elements of A. Then for each $p \in Parents(A)$, we check if the configurations that have some $a \in A$ cover exactly the configurations that have p (line 5). If the configuration sets associated with each a form a partition (pairwise disjoint), A is a Xor-group with parent p of size $|A|$ (lines 7-8). If they do not form a partition, and A does not include a smaller Xor- or Or-group (lines 11-12), A is an Or-group with parent p of size $|A|$ (lines 13-14).

Algorithm 2: ComputeXorOrGroups(K)

```
   Data: Formal context K, AC-poset (AC_K, ⩽_s)
   Result: XorGroups, OrGroups
1  XorGroups = ∅; OrGroups = ∅;
2  foreach i ∈ {2..|AC_K|} do
3      foreach Antichain A of size i in (AC_K, ⩽_s) do
4          foreach p ∈ Parents(A) do
5              if p' = ⋃_{a∈A} a' then
6                  if ∀a1, a2 ∈ A, a1' ∩ a2' = ∅ then
7                      XorGroups+ = (A, p);
8                  end
9                  else
10                     if ∀xor ∈ XorGroups, xor ⊄ A,
11                     and ∀or ∈ OrGroups, or ⊄ A then
12                         OrGroups+ = (A, p);
13                     end
14                 end
15             end
16         end
17     end
18 end
```

In contrast with the proposal of [9], we do not apply heuristics to extract a tree structure and implication rules to conform with the FM formalism. This part of their work (which suits their objective) creates structures that are no more canonical. We keep the whole directed acyclic graph.

The last part of our approach takes another track and consists in building the Mutex (Algorithm 3), that will encode the missing knowledge about the exclusions between features. A Mutex is a set of mutually exclusive features, whose associated configurations do not cover the configurations of a specific feature (i.e. they have no parent). Mutex are extracted by growing size (line 2-3). A feature set is a Mutex if the features have no configuration in common, the set does not contain a Xor-group, is not included in a Xor-group (line 4) and, either its size is 2, or it does not contain any smaller Mutex (line 5).

Algorithm 2 is exponential, as explained in [9], but the authors also mention that input size is $|O| \times |A|$, and they argue that the computation can be reasonable in most cases. The same complexity arguments apply to Algorithm 3. Besides, they can be implemented efficiently (although still with exponential complexity) by avoiding generating useless antichains, namely, those that contain previously recognized groups of some categories (lines 11-12 in Algorithm 2, lines 4-5 in Algorithm 3). In Algorithm 2, this corresponds to the min-set-cover problem solving as mentioned in [9].

Algorithm 3: ComputeMutex(K)

```
   Data: Formal context K, AC-poset (AC_K, ⩽_s)
   Result: Mutex
1  Mutex = ∅;
2  foreach i ∈ {2..|AC_K|} do
3      foreach Antichain A of size i ∈ (AC_K, ⩽_s) s.t.
       ⋂_{a∈A} a' = ∅ do
4          if ∀xor ∈ XorGroups, (A ⊄ xor and xor ⊄ A)
5          and (|A| = 2 or ∀m ∈ Mutex, m ⊄ A) then
6              Mutex+ = A;
7          end
8      end
9  end
```

4. Study of PCMs variability structure

In this section, we show the application of our approach to a set of real PCMs and a few examples of the roles the ECFD can play for data summary, for FM extraction or for product selection.

Dataset description We selected 21 PCMs from wikipedia, that illustrate typical cases. These PCMs describe software products such as: wikis, data-base editors, media players, licenses or web browsers. They have been cleaned as described in [2] and transformed into Formal Contexts[2] with scaling operations [5]. Pages of wikipedia product descriptions are sometimes split in several PCMs. Some products even do not appear in all PCMs. In these cases, we kept this splitting, rather than concatenating the PCMs, because we think that it represents some domain knowledge.

ECFD computation and description The AC-posets are built with RCAexplore[3], and the groups with a specific Java program. Table 2 presents figures about the selected PCMs. Due to the scaling operations, a feature is a pair (property, value). The number of described products (`#conf`) varies between 9 and 90. The number of (property, value) pairs (`#feat`) varies between 5 and 67.

[2]Available with the corresponding AC-posets at: `www.lirmm.fr/recherche/equipes/marel/datasets/fca-and-pcm`

[3]http://dolques.free.fr/rcaexplore/

Table 2. Figures for TangEshop FM and for the selected 21 PCMs from our dataset. The PCMs are organized by size and topic. The number placed against the PCM name is its identifier in the dataset.

PCM name (id)	$\|\#conf.\| \times \|\#feat.\|$	ECFD Groups				ECFD structure	
		#xor (nb,size)*	**#or (nb,size)***	**#mutex (nb,size)***	**#mutex** of size 2	**#nodes**	**#multi-parents (nb,size)***
TangEshop	13 × 10	1 (1:2)	2 (2:2)	1 (1:2)	1	7	1 (1:2)
MathSoft (19)	9 × 8	0	1 (1:2)	0	0	5	1 (1:2)
Dbedition (5)	22 × 5	0	0	0	0	4	0
PublicLicense (27)	12 × 16	1 (1:3)	1 (1:3)	7 (6:2)(1:3)	6	10	2 (2:2)
Wiki (2)	40 × 7	0	1 (1:2)	0	0	7	1 (1:4)
Wiki (3)	38 × 6	0	4 (3:2)(1:3)	0	0	5	0
3Dsoft (7)	36 × 14	0	0	18 (18:2)	18	11	4 (3:2)(1:3)
Player (11)	33 × 9	0	0	13 (13:2)	13	9	0
Music (12)	25 × 8	0	0	0	0	8	2 (2:2)
Mail (39)	16 × 16	1 (1:2)	24 (13:2)(11:3)	7 (7:2)	7	14	5 (3:3)(1:4)(1:5)
PublicLicense (26)	51 × 6	0	0	0	0	6	0
Texteditor (17)	66 × 5	0	0	0	0	5	0
Web browser (21)	43 × 9	0	0	0	0	9	4 (1:4)
Download (15)	34 × 27	1 (2:1)	0	180 (171:2)(9:3)	171	24	9 (2:2)(7:3)
LinuxAdr (20)	34 × 16	0	0	5 (5:2)	5	13	6(2:3)(4:2)
Webbrowser (23)	36 × 12	0	0	2 (2:3)	0	12	2 (1:4)(1:5)
Webbrowser (22)	41 × 13	1 (1:2)	0	3 (3:2)	3	12	4 (1:4)
Prog. Languages (31)	56 × 25	0	0	212 (93:2) (88:3)(27:4)(4:5)	93	25	10 (4:2)(3:3) (2:4)(1:5)
Prog. Languages (32)	40 × 10	0	0	26 (8:2)(18:3)	8	10	1(1:3)
Prog. Languages (33)	90 × 10	0	0	17 (15:2)(2:3)	15	10	1 (1:4)
SocialNetJour (29)	11 × 67	70 (11:2)(1:3)(2:4)(6:5) (20:6)(21:7)(8:8)(1:9)	95 (1:4)(4:5)(43:6) (40:7)(7:8)	104 (96:2)(8:3)	96	25	11 (6:2)(5:3)
HtmlRendering (24)	15 × 61	15 (4:2)(1:3)(3:4) (4:5)(1:6)(1:7)(1:8)	3 (1:4)(2:5)	135 (135:2)	135	22	8 (8:2)

`#xor`, `#or` and `#mutex` are respectively the number of Xor-groups, Or-groups and Mutex. The number of Mutex of size 2 is isolated in one column to show the mutex that are commonly used in FM formalisms. `#nodes` is the number of ECFD nodes, and also the number of AC-poset nodes (and the number of feature equivalence classes). `#multi-parents` is the number of ECFD nodes that have more than one direct parent. `#xor`, `#or`, `#mutex` and `#multi-parents` are followed by a list of `(group-size:group-number)` pairs. For example, `24 (13:2)(11:3)`, in PCM Mail-39 `#or` groups means that in the corresponding ECFD, there are 24 Or-groups, including 13 groups of size 2, and 11 groups of size 3.

ECFD as a data summary The ECFD helps to identify the situations where many products have the same description and are grouped together (PCMs 2, 3, 23, 33); or inversely where product groups are smaller (PCM 15). The AC-poset shows feature sets that few products have (such as in PCM 7), revealing possibilities for increasing the product range, or balancing the product offer (if the products are developed together or belong to a same brand). A complex ECFD (PCMs 24, 29) should alert a designer to simplify, or refine the product offer. She/he also should check the cases where the ECFD is not complex, except for the number of Mutex (PCM 15, 31). We observe that beyond 20 (property, value) pairs, the number of Mutex becomes large (up to 212) which is specific to ECFD coming from PCMs (it does not appear for ECFD built upon the configuration set of an existing FM, as we studied in [3] in the context of FM composition).

ECFD for FM extraction We believe that the ECFD is a good candidate for FM extraction. When the multi-parent number is null or low (such as in PCMs 26, 17, 32, 33), the FM tree should be easy to extract. Furthermore, if there are no Mutex (such as in PCMs 26, 17), the FM will not include additional constraints and will be more intuitive. PCM 29, in the opposite case, has many multi-parents and many Mutex and will be difficult to transform into an FM. When the AC-poset has several roots (as this is the case for PCMs 24 or 29), they can be used to structurally decompose the description into several FMs, using also information about the semantics of the features, as recommended by methodologies like OVM [8]. Besides, the ECFD, which contains all the possible FMs, is a relevant structure to support a methodology which guides a designer through the FM extraction process. Mutex of size 2 can easily be used in a translation from the ECFD to an FM. In the studied PCMs, they cover the largest part of the Mutex set. During the FM extraction process, the FM should cover the products present in the PCM, but should not be too strict, thus it may

accept products designed in (or suggested for) the future, and e.g., may not take into account the numerous Mutex.

ECFD for product selection Conceptual structures are good candidates for free datasets navigation, as shown in [11, 6]. In [11, 6], the conceptual structures contain both objects and properties, while the ECFD only contains the properties (here features). However, the ECFD still is a navigation structure, constrained by the groups, which allows to select products and, while doing this selection, be able to switch to close products, more easily than in a tree structure, by using the AC-poset relationships, especially in complex cases like PCMs 24, 29.

5. Related Work

To our knowledge, besides FM, there are two other graphical representations which can depict the variability of a set of product descriptions. A *Feature Graph* [10] is a diagram-like representation of several FMs describing the same configurations. It is composed of a directed acyclic graph representing implications between features, and a set of textual lists of feature-groups (*Or*, *Xor* and *Mutex*). The feature graph transitive closure/reduction is canonical, but it may describe more configurations than in the initial set of products. Another possible graphical representation is the *Binary Implication Graph* [4], representing all feature implications that can be extracted from a set of product descriptions. The potential feature-groups are not documented. As for the feature graph, only the transitive closure/reduction is unique, and some sets of products cannot be strictly represented with this formalism. By comparison, the ECFD is a canonical and full graphical representation of the scope of a PCM. We made a recapitulative table of the different graphical representations[4].

As already mentioned in the paper, Ryssel et al. [9] have also worked on extracting variability information (e.g., FM hierarchy, feature groups) from a set of product descriptions using FCA. However, while they focus on building one FM, we keep all variability information extracted from conceptual structures to represent, in the ECFD, all possible FMs depicting the initial products. Finally, Acher et al. propose an approach to extract FMs from PCMs based on FM merged operations [1]. Here again, they focus on building one consistent FM that best depicts the PCM, and not a unique structure preserving all knowledge about the PCM variability. However, if building an FM is the last step of a design, the ECFD can be considered as an interim step.

[4] http://www.lirmm.fr/recherche/equipes/marel/datasets/variability-representation

6. Conclusion

In this paper, we proposed the ECFD structure which gives a graphical and canonical view on variability. We applied it to software PCMs taken from wikipedia and showed its use for data analysis, product selection or as a guide for building FMs. The experimentation results are heterogeneous and show the necessity to deepen the analysis of PCMs.

As future work, we would like to evaluate our approach against other existing ones. We plan to apply our ECFD construction to PCMs of other domains and to define a methodology for guiding the designer from an ECFD to its alternative FM representation. This should include a decomposition step, where the ECFD will be relevant. We also want to study the situations where several PCMs are connected, like a PCM of multimedia softwares connected to a PCM of databases.

References

[1] M. Acher, A. Cleve, G. Perrouin, P. Heymans, C. Vanbeneden, P. Collet, and P. Lahire. On extracting feature models from product descriptions. In *6th Int. VaMoS*, pages 45–54, 2012.

[2] J. Carbonnel, M. Huchard, and A. Gutierrez. Variability representation in product lines using concept lattices: Feasibility study with descriptions from wikipedia's product comparison matrices. In *Int. Works. FCA&A @ ICFCA*, pages 93–108, 2015.

[3] J. Carbonnel, M. Huchard, A. Miralles, and C. Nebut. Feature Model composition assisted by Formal Concept Analysis. In *12th Int. Conf. ENASE*, pages 27–37, 2017.

[4] K. Czarnecki and A. Wasowski. Feature Diagrams and Logics: There and Back Again. In *11th Int. Conf. on Soft. Product Lines (SPLC)*, pages 23–34, 2007.

[5] B. Ganter and R. Wille. *Formal concept analysis - mathematical foundations*. Springer, 1999.

[6] G. J. Greene and B. Fischer. Single-focus broadening navigation in concept lattices. In *Proceedings of the 3rd Works. CDUD @ (CLA 2016)*, pages 32–43, 2016.

[7] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-Oriented Domain Analysis (FODA): Feasibility Study. *Technical Report CMU/SEI-90-TR-21 - ESD-90-TR-222*, 1990.

[8] K. Pohl, G. Böckle, and F. J. van der Linden. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer Science & Business Media, 2005.

[9] U. Ryssel, J. Ploennigs, and K. Kabitzsch. Extraction of feature models from formal contexts. In *15th (SPLC) Workshop Proceedings (Vol. 2)*, page 4, 2011.

[10] S. She, U. Ryssel, N. Andersen, A. Wasowski, and K. Czarnecki. Efficient synthesis of feature models. *Information & Software Technology*, 56(9):1122–1143, 2014.

[11] T. Wray and P. W. Eklund. Exploring the information space of cultural collections using formal concept analysis. In *9th Int. Conf. ICFCA*, pages 251–266, 2011.

An Empirical Study on the Equivalence and Stability of Feature Selection for Noisy Software Defect Data

Zhou Xu[1], Jin Liu[1]*, Zhen Xia[1], Peipei Yuan[2]
[1]State Key Laboratory of Software Engineering, School of Computer, Wuhan University, Wuhan, China
[2]School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China
*Corresponding author email: jinliu@whu.edu.cn

***Abstract*—Software Defect Data (SDD) are used to build defect prediction models for software quality assurance. Existing work employs feature selection to eliminate irrelevant features in the data to improve prediction performance. Previous studies have shown that different feature selection methods do not always yield similar prediction performance on SDD, which indicates that these methods are not equivalent. Also, previous studies have shown that SDD usually contains noise that may interfere the process of feature selection. In this work, we empirically investigate and measure the equivalence of different feature selection methods for SDD. Further, we intend to analyze the stability of the methods for noisy SDD. We perform statistical analyses on eight projects from NASA dataset with eight feature selection methods. For the equivalence analysis, we introduce Principal Component Analysis (PCA) and overlap index to qualitatively and quantitatively analyze the equivalence of these methods respectively. For the stability analysis, we apply consistency index to measure the stability of these methods. Experimental results indicate that different feature selection methods are indeed not equivalent to each other, and Correlation and Fisher Score methods achieve better stability.**

Keywords—defect data; feature selection; equivalence analysis; stability analysis;

I. INTRODUCTION

Software debugging is a key and expensive phase during the software development lifecycle. Defect identification is one of the main activities for software debugging [1], [2]. Software Defect Prediction (SDP) automatically detect the more defect-prone software modules (methods, classes or files) with software metrics (i.e., features) in Software Defect Data (SDD).

In the past decade, many researchers mainly focused on applying various data mining and machine learning algorithms to build defect prediction models on SDD for SDP, to identify the quality of a given software module by classifying it as defect-prone or not [3], [4], [5]. Defect prediction can help practitioners to reasonably allocate limited project resources to the potential defect-prone modules, and thus improve the efficiency and save the cost of software development.

One challenge in defect prediction modeling is the high dimensionality phenomenon, i.e. there may exist irrelevant or redundant features in SDD. Building prediction models with all features is unrealistic since these useless features may deteriorate the performance of prediction models. Thus, the issue of how to select the most appropriate features is of great importance.

Nowadays, a plenty of feature selection methods have been proposed in data mining area. Various methods have been successfully introduced to assist the selection of a feature subset that could benefit the defect prediction process on SDD. Previous studies have shown that diverse feature selection methods yield quite different performance on prediction models for SDD [6], [7], which implies that different methods might be not equivalent, that is, different methods would identify different set of features as relevant. However, to the best of our knowledge, no previous studies proposed a method to investigate the equivalence of different feature selection methods. In this paper, we conduct an empirical study to qualitatively analyze whether different feature selection methods are equivalent with Principal Component Analysis (PCA) technique. Further, we use overlap index to quantitatively analyze to what extent different feature selection methods are not equivalent each other.

Meanwhile, due to some unexpected reasons, such as improper software data collection and recording process [8], there may exist noise in SDD. One potential challenge for feature selection methods is their sensitivity to the noise in SDD (i.e. the impact of noise on the selection of relevant features from SDD), which is defined as the stability of the feature selection methods to the noise on SDD in this paper. It is valuable to study the stability of different feature selection methods, because if the features selected by a specific method vary when there exists noise in the dataset, it is difficult to say whether these features are the most representative ones. If a feature selection method is high in stability, it will enable practitioners to select the representative features of SDD before cleaning the dataset. However, there are few relevant studies on this issue yet. In this work, we employ consistency index to measure the stability of feature selection methods on noisy SDD.

Statistical analyses on the data from eight projects of NASA dataset confirm that different feature selection methods are not equivalent to each other. The analytic results suggest that Correlation (Cor) and Fisher Score (FS) methods are more stable.

Our main contributions are highlighted as follows:

(1) We introduce Principal Component Analysis (PCA) technique to investigate the equivalence of different feature selection methods. To the best of our knowledge, this is the first empirical study to qualitatively analyze the equivalence of feature selection methods.

(2) We employ overlap index to measure to what extent these feature selection methods are not equivalent to each other.

(3) We analyze the stability of feature selection methods in the context of noisy SDD and identify the stable ones.

DOI reference number: 10.18293/SEKE2017-097

II. Related Work

A. Feature Selection on SDD

Many previous studies have investigated the effect of feature selection on the performance of defect prediction models. Song et al. [3] pointed out that feature selection is an indispensable part of a general defect prediction framework. Shivaji et al. [9] investigate the impact of six methods on the classification-based bug prediction. They found that selecting about 10% features could achieve a satisfactory performance. He et al. [21] suggested that prediction models built with a simplified feature set could achieve acceptable performance for defect prediction.

All these studies only focus on evaluating different feature selection methods using prediction performance indicators, none of them pay attention to the equivalence between these methods on their suggested features. In this work, we conduct an empirical study to investigate this issue and also evaluate feature selection methods from the perspective of stability.

B. Stability of Feature Selection for SDD

Recently, a few studies evaluated feature selection methods by their stability for SDD. The stability of a method is defined as the degree of consensus of a feature subset pair selected by the method on two variants of a given dataset obtained by sampling. It is worth studying the stability of feature selection methods since the method that tends to select the highly similar features despite changes in the data could be more trustworthy.

To the best of our knowledge, the earliest study on the stability of the feature selection methods for SDD was performed by Gao et al. [26]. They empirically studied the impact of three data sampling techniques on the stability of six filter-based feature selection methods for SDD. Wang et al. [22] investigated the impact of the dataset perturbations (i.e. randomly removing a certain proportion software modules) and the number of selected features on 6 filter-based feature selection methods for SDD.

Different from our empirical study, these literatures aim to explore the stability of feature selection methods to the data perturbation on SDD, while our work focuses on investigating the stability of the methods to the noise on SDD. To the best of our knowledge, this is the first work to investigate this issue.

III. Preliminary and Analysis Method

In this work, we conduct an empirical study to investigate two issues: the equivalence analysis to explore whether different feature selection methods select similar features for SDD and stability analysis to investigate the stability of different feature selection methods for noisy SDD.

A. Feature Ranking Methods

Feature selection is a critical data preprocessing technique in many fields. In general, the feature selection methods fall into two major categories as feature ranking and feature subset selection [10], [25]. The feature ranking methods have gained favor due to its simplicity and efficiency. In this paper, we empirically study the equivalence and stability of feature ranking methods for noisy SDD. The eight methods used in this work are Chi-Square (**CS**), Correlation (**Cor**), Information Gain (**IG**), Symmetrical Uncertainty (**SU**), Fisher Score (**FS**), Welch T-Statistic (**WTS**), ReliefF (**RF**), One Rule (**OneR**). The reason why we choose these methods is that they are widely used in defect prediction and belong to different feature selection families [28], [30]. CS is a statistic-based method, Cor is a correlation-based method, IG and SU are entropy-based methods, FS and WTS are first order statistics-based methods, RF is a instance-based, OneR is a classifier-based method. The detailed description of these methods are available in [11], [29].

B. Research Questions

To reveal the answers of the two issues, we empirically study the following three research questions (RQs).

RQ1: Are different feature selection methods equivalent to each other when being applied to SDD?

RQ2: To what extent different feature selection methods are equivalent to each other?

RQ3: Which feature selection method is more stable among the six methods for noisy SDD given in this study?

C. Equivalence Analysis

The first two research questions aim to conduct the equivalence analysis of feature selection methods. In this work, we introduce PCA technique to qualitatively analyze the equivalence of the eight methods (***RQ1***). Further, we apply the overlap index to quantitatively analyze to what extent these methods are equivalent to each other (***RQ2***).

1) PCA

We assume that given n evaluated objects (i.e. n features in this work), let $x_1, x_2, \dots, x_m$ denote the m indicator variables (i.e. m different feature selection methods in this work) and $x_{ij}, (i = 1,2, \dots, n; j = 1,2, \dots, m)$ denotes the score of the jth evaluated object with the ith indicator variable, namely the relevant score of the ith feature assigned by the jth feature selection method. PCA technique involves four steps:

a) Normalization

Considering the difference in score values assigned by different indicator variables, we apply the z-score to normalize the values in this step. The x_{ij} can be transformed to $\tilde{x}_{ij}$ as:

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j} \tag{1}$$

where $\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$, $s_j = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2}$.

b) Correlation coefficient matrix

Let $R = (r_{ij})_{m\times m}$ denotes the correlation coefficient matrix with respect to the indicator variables, where r_{ij} denotes the correlation coefficient between the ith and the jth indicator variables. r_{ij} $(i, j = 1,2, \dots, m)$ is defined as:

$$r_{ij} = \frac{\sum_{k=1}^{n} \tilde{x}_{ki}\tilde{x}_{kj}}{n-1} = \frac{\sum_{k=1}^{n}(x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j)}{\sqrt{\sum_{k=1}^{n}(x_{ki} - \bar{x}_i)^2 \cdot \sum_{k=1}^{n}(x_{kj} - \bar{x}_j)^2}} \tag{2}$$

c) Eigenvalues and eigenvectors

In this step, we calculate the eigenvalues and eigenvectors of the correlation coefficient matrix R. By solving the determinant $|R - \lambda I| = 0$ (I is identity matrix), we can obtain m eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_m$,$(\lambda_1 \geq \lambda_2 \geq, \dots, \geq \lambda_m \geq 0)$ and the corresponding eigenvectors $\alpha_1, \alpha_2, \dots, \alpha_m$, where $\alpha_i = [\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{im}]^T (i = 1,2, \dots, m)$. Then the original indicator variables are projected to m new orthogonal variables as:

$$\begin{cases} y_1 = \alpha_{11}x_1 + \alpha_{12}x_2 + \cdots + \alpha_{1m}x_m \\ y_2 = \alpha_{21}x_1 + \alpha_{22}x_2 + \cdots + \alpha_{2m}x_m \\ \vdots \\ y_m = \alpha_{m1}x_1 + \alpha_{m2}x_2 + \cdots + \alpha_{mm}x_m \end{cases} \tag{3}$$

where $\alpha_{k1}^2 + \alpha_{k2}^2 + \cdots + \alpha_{km}^2 = 1$ ($k = 1,2, \ldots, m$), y_i denotes the i th principal component, and α_{ij} denotes the correlation coefficient between the original variable x_j and the principal component y_i. The coefficient α_{ij} represents the contribution of the original variables x_j to the principal component y_i, i.e., the importance of x_j to y_i.

In this work, we use this coefficient to measure the correlation between different feature selection methods and the principal components. A greater absolute value of the correlation coefficient α_{ij} indicates that the j th original variable captures the ith principal component well. If different feature selection methods capture same principal components, it indicates that these methods are equivalent to each other.

d) Contribution percentage

Let c_i denotes the percentage of the variance of principal component y_i in total variances, which is also called the contribution percentage of the principal component y_i to original variables [12]. It is defined as:

$$c_i = \frac{\lambda_i}{\sum_{k=1}^{m} \lambda_k} \tag{4}$$

where λ_i denotes the i th eigenvalue of the correlation coefficient matrix R. The contribution percentages reflect the synthesis or explanatory ability of the principal components towards the original variables. The first principal component is the largest and the rest can be deduced by analogy.

Then the cumulative contribution percentage cc_p of the top p principal components is defined as:

$$cc_p = \frac{\sum_{i=1}^{p} \lambda_i}{\sum_{j=1}^{m} \lambda_j} \tag{5}$$

where $p \leq m$. This index reflects the synthesis ability of the top p principal components towards the original variables.

In this work, we use the contribution percentage to measure the percentage (or the amount of information) of the original variables covered by different principal component.

2) Overlap Index

The analysis based on PCA enables us to test whether different feature selection methods are equivalent or not. It is unknown yet to what extent these methods are equivalent to each other. We further use an indicator, called overlap index [24], to measure the extent of equivalence between any method pair.

In this work, we apply overlap index to quantify the extent of the equivalence under a cut point μ. Note that μ is actually a pre-specified percentage of selected features. Given a specific cut point μ, we compute the following overlap index as:

$$|(A \cap B)_\mu| = |D(A) \cap D(B)|_\mu \tag{6}$$

where $D(A)$ and $D(B)$ denote the top μ relevant features selected by feature selection method A and B for a specific dataset D, respectively. Under the cut point μ, $|(A \cap B)_\mu|$ denotes the cardinal number of features selected by both methods. In general, a higher overlap value indicates a larger extent of equivalence of the two methods under a given cut point μ. In this work ,we empirically set μ as 15%, 20%, 25%, 30%, 35% and 40%.

D. Stability Analysis

A potential threat for feature selection methods is the presence of noise in the data. It would be valuable to study of the stability of different feature selection methods for noisy dataset. If different feature selection methods are not equivalent to each other, the identification of the most stable method will benefit the selection of the high representative features of the noisy data without cleaning it.

In this work, we further conduct an empirical study to investigate the stability of the eight methods and attempt to find out the more stable ones for noisy SDD (***RQ3***). Specifically, we use an indicator to compute the similarity of the relevant feature subset pair selected from the noisy and clean versions of a dataset with a feature selection method.

Previous studies have proposed different similarity metrics to measure the stability of feature selection methods [13]. In this study, we employ consistency index [14]. Given a feature selection method A, dataset D and cut point μ, let U_1 and U_2 denotes the feature subsets selected by method A from noisy and clean versions of D, respectively. Then consistency index $IC(A)_{D_\mu}$ is defined as:

$$IC(A)_{D_\mu} = \frac{dn - t2}{t(t-n)} \tag{7}$$

where d denotes the cardinality of the intersection of U_1 and U_2, n denotes the total number of the features in D, t denotes the cardinality of U_1 or U_2, i.e., the number of selected features, and $-1 < IC(A)_{D_\mu} \leq 1$. Consistency index have been widely used to measure the stability of feature selection methods to the random perturbation on the dataset [15], [16].

IV. BENCHMARK DATASET

To investigate the equivalence and the stability of the feature selection methods for noisy SDD, we used eight original version projects of NASA dataset and the corresponding clean version preprocessed by Shepperd et al. [17] as our experimental dataset. NASA dataset is a method-level software defect dataset that is characterized by static code metrics [5]. The original NASA dataset is known to be noisy, to alleviate the data quality of this dataset, Shepperd et al. applied some preprocessing criteria to clean the dataset. We remove the features with one value. Table I presents the details of the two versions of the eight projects, including the number of features (#F), modules (#M) defective modules (#D) and the percentage of defective modules (%D).

TABLE I. DESTRIPTION OF THE TWO VERSIONS OF NASA DATASET

Projects	#F	Noise			Clean		
		#M	#D	%D	#M	#D	%D
CM1	37	505	48	9.5%	327	42	12.8%
KC1	21	2107	325	15.4%	1162	294	25.3%
KC3	39	458	43	9.4%	194	36	18.6%
MC1	38	9466	68	0.7%	1847	36	2.0%
MC2	39	161	52	32.3%	125	44	35.2%
MW1	37	403	31	7.7%	251	25	10.0%
PC1	36	5589	23	0.4%	734	16	2.2%
PC5	38	17186	516	3.0%	1679	459	27.3%

V. ANALYSIS RESULTS

A. *RQ1*

To answer this question, we conduct the analyses with the eight feature selection methods on the data from eight projects of original NASA dataset with PCA technique. Owing to space constraint, Table II and III only report the analytic results on KC3 and PC5 projects randomly selecting from the eight studied projects. The Ci ($i = 1, \ldots, 8$) denotes the i th principal

TABLE II. RESULTS OF PCA ON KC3 PROJECT

Methods	C1	C2	C3	C4	C5	C6	C7	C8
CS	**0.40**	-0.27	0.12	-0.23	0.51	-0.62	-0.22	0.03
Cor	**0.42**	-0.13	-0.09	0.28	-0.32	-0.05	0.10	0.78
IG	**0.42**	-0.20	-0.14	-0.13	0.33	0.75	-0.29	-0.01
SU	**0.41**	-0.19	-0.21	-0.24	-0.16	-0.01	0.73	-0.36
FS	**0.41**	0.00	0.08	0.48	-0.40	-0.11	-0.39	-0.51
WTS	0.25	**0.55**	0.42	0.38	0.43	0.09	0.34	0.01
RF	-0.27	-0.50	-0.29	**0.65**	0.36	-0.02	0.19	-0.08
OneR	-0.10	-0.53	**0.80**	-0.03	-0.15	0.17	0.11	0.01
CP(%)	61.19	18.04	10.37	5.60	2.33	0.98	0.82	0.67

TABLE III. RESULTS OF PCA ON PC5 PROJECT

Methods	C1	C2	C3	C4	C5	C6	C7	C8
CS	0.34	0.55	0.07	0.07	0.27	0.01	-0.06	0.70
Cor	**0.49**	-0.06	-0.25	-0.22	-0.30	-0.15	0.73	0.03
IG	0.28	**0.63**	0.05	-0.05	0.14	0.15	-0.01	-0.69
SU	0.33	-0.07	-0.05	**0.86**	-0.35	0.04	-0.12	-0.06
FS	**0.46**	-0.13	-0.28	-0.40	-0.26	-0.16	-0.67	0.00
WTS	0.30	-0.26	**0.59**	-0.19	-0.12	0.66	0.01	0.06
RF	-0.32	0.27	-0.52	-0.07	-0.39	0.61	-0.01	0.14
OneR	0.24	-0.36	-0.48	0.11	0.67	0.34	0.03	-0.06
CP(%)	42.82	22.26	14.72	9.02	7.24	3.55	0.27	0.13

component, the CP denotes the contribution percentage while the other values represent the correlation coefficients. For the CP values, from the complete PCA results on all projects (see in [32]), we observe that PCA technique can mainly identify four principal components with a cumulative contribution percentage between 88.82% (for PC5) and 97.45% (for KC1) to capture the most information of the original variables (i.e., feature set in this study). The first principal component contributes the major proportion varying from 42.73% to 74.67%, and the second principal component contributes the proportion varying from 12.12% to 22.26%, while the third and the forth components contribute proportion varying from 6.06% to 18.27% and from 3.28% to 7.03%, respectively.

For the correlation coefficients between the methods and the principal components, the values in bold highlight the feature selection methods that best represent the corresponding principal components. Note that the negative coefficients only denote negative correlation without the meaning of numerical size. From Table II, we observe that, for KC3 project, the first component, which reflects 61.19% amount of information towards the original feature set, is mainly captured by CS, Cor, IG, SU, and FS (since their coefficient values are very close) while poorly captured by the other methods; the second principal component is only captured by WTS; the third principal component is mainly captured by OneR as the coefficient value 0.8 and the fourth principal component is mainly captured by RF as the coefficient value 0.65. Table III shows that, for PC5 project, the first component is mainly captured by Cor and FS while poorly captured by the others; the second to the fourth principal component are captured by IG, SU, WTS, respectively. For other projects (see in [32]), similar observations are obtained except that the methods that capture each component vary among different projects.

As mentioned above, the analytic results of the PCA indicate that different feature selection methods indeed capture different components. It confirms that different feature selection methods do not have the similar effect on selecting the relevant features, i.e., they assign different relevance proneness to the module features. Besides, for the methods that belong to the same feature selection family (i.e., IG and SU, FS and WTS), they are not always capture the same principal component. For example, on KC3 project, FS captures the first principal component while WTS captures the second component; on PC5 project, IG and SU capture the second and the fourth principal component respectively while FS and WTS capture the first and the third component respectively. It denotes that the feature selection methods that belong to the same family are not equivalent to select the similar relevant features. In addition, we find that Cor and FS capture the same principal component on most projects.

> We conclude that, in general, different feature selection methods be not equivalent to each other for the given SDD in this study.

B. *RQ2*

Although different feature selection methods are testified to be not equivalent to each other by PCA in Section V-A, but it could not tell us to what extent they are equivalent to each other. This question is actually to supplement this issue using the overlap index.

We also only report the analysis results on PC5 project in Table IV due to the space limit (see in [32] for complete results). The last line of the table reports the average percentage (AP) of the overlap index. For example, for CS and Cor pair, the overlap value is equal to 1 while the number of the selected features is equal to 6 under the cut point 15%, so the percentage of the overlap is equal to 16.7% (1/6). From the table, we observe that the overlap values of most method pairs are relative low. For example, most overlap values are equal to 0 under the cut point 15%. In addition, most overlap percentages are less than 50% under four cut points. But for the method pair Cor and FS, the overlap values are nearly equal to the total number of selected

TABLE IV. OVERLAP RESULTS ON PC5 PROJECT

A	B	15%	20%	25%	30%	35%	40%
CS	Cor	1	1	2	2	3	6
CS	IG	3	5	8	11	13	14
CS	SU	0	0	0	1	1	3
CS	FS	1	1	2	2	3	6
CS	WTS	1	1	2	2	5	7
CS	RF	0	0	0	2	4	7
CS	OneR	0	0	0	0	2	6
Cor	IG	0	0	1	2	4	5
Cor	SU	0	0	2	5	6	8
Cor	FS	6	8	10	12	14	16
Cor	WTS	3	3	6	6	8	12
Cor	RF	1	2	3	4	5	7
Cor	OneR	3	4	5	7	8	11
IG	SU	0	0	0	1	1	2
IG	FS	0	0	1	2	4	5
IG	WTS	0	1	1	2	5	6
IG	RF	0	0	0	2	4	6
IG	OneR	0	0	0	0	1	4
SU	FS	0	0	2	5	6	8
SU	WTS	0	0	2	3	4	6
SU	RF	0	0	0	0	3	3
SU	OneR	0	3	6	8	9	10
FS	WTS	3	3	6	6	8	12
FS	RF	1	2	3	4	5	7
FS	OneR	3	4	5	7	8	11
WTS	RF	3	4	4	5	8	9
WTS	OneR	1	1	2	5	5	9
RF	OneR	1	1	1	1	2	4
AP(%)		18.5	19.6	26.4	31.8	38.0	46.9

features under all cut points. This observation accords to that in ***RQ1*** which shows that the Cor and FS usually capture the same principal component. For other projects (see in [32]), we can also get the similar observations.

Moreover, from the complete results in [32], we find that the overlap percentage increases as the cut point value increases on most cases. The reason may be that a feature selection method pair tends to select more common features as the relevant ones when the cut point increases.

> To sum up, the analytic results show that the overlap values and their average percentages of method pairs are low on most case. This observation also confirms that different feature selection methods are quite not equivalent to each other.

C. RQ3

Since different feature selection methods are indeed not equivalent (as shown in ***RQ1*** and ***RQ2***) and the noise is inevitable in SDD, in this question, we are particularly interested in investigating the stability of feature selection methods for noisy SDD. As the first work to empirically study this issue, we conduct the analyses with the eight feature selection methods on the two versions of eight NASA projects. We apply consistency index to measure the stability of these methods.

TABLE V. AVERAGE STABILITY ON EACH PROJECT UNDER SIX CUT POINTS

Cutoff	CS	Cor	IG	SU	FS	WTS	RF	OneR
15%	0.54	0.83	0.44	0.38	0.83	0.67	0.66	0.50
20%	0.55	0.80	0.47	0.33	0.80	0.71	0.74	0.42
25%	0.52	0.85	0.54	0.35	0.83	0.67	0.78	0.50
30%	0.50	0.84	0.58	0.37	0.84	0.72	0.70	0.51
35%	0.59	0.85	0.56	0.41	0.83	0.73	0.72	0.55
40%	0.60	0.80	0.53	0.40	0.77	0.79	0.76	0.58

Table V summaries the average stability values of each method across all projects under each cut point (detailed results is available in [32]). We observe that Cor and FS methods can achieve better results under all cut points, and their values are very close. It is also consistent with the observations in ***RQ1*** and ***RQ2*** that Cor and FS select very similar features. In addition, WTS and RF also achieve competitive results under the cut point 40%.

TABLE VI. P-VALUES AND AVERAGE RANKINGS UNDER EACH CUT POINTS

Cutoff	p-values	CS	Cor	IG	SU	FS	WTS	RF	OneR
15%	0.0057	4.94	2.75	5.50	6.25	2.31	4.38	4.31	5.56
20%	0.0002	4.69	2.69	5.81	6.56	2.44	3.63	3.63	6.56
25%	0.0012	5.56	2.38	5.69	6.69	2.88	4.31	3.13	5.38
30%	0.0004	6.06	2.63	4.81	6.56	2.63	3.19	4.25	5.88
35%	0.0016	5.00	2.88	5.06	6.81	2.38	3.38	4.63	5.88
40%	0.0080	4.94	3.25	5.38	6.94	3.56	3.06	3.50	5.38

To statistically analyze the stability results under each cut point, we perform Friedman test, a non-parametric test, to compare the eight methods over the eight projects by ranking each method on each project separately. The p-value less than 0.05 in Friedman test indicates that the average rankings of these methods are statistically significant. We further apply Nemenyi test as a post-hoc test to all pairs of methods to determine which method performs statistically different.

Table VI reports the p-values of Friedman test and the average ranking of each method across all projects under each cut point. The p-values of Friedman test in the table are all less than 0.05, which indicates that the differences among the average rankings of these methods are significant under each cut point. In addition, the lower average ranking value represents the higher stability. We observe that Cor and FS usually obtain better ranking values. Fig.1 visualizes the results of Nemenyi test in terms of stability index. In the figure, the average rankings of the methods are plotted on the top line in each subfigure under each cut point. Methods that are not statistically significant are connected with blue lines. The lower average ranking locates on the left side of the axis.

This figure depicts that the stability results between FS and SU under cut point 15%, between FS, Cor and OneR, SU under cut point 20%, between Cor, FS and SU under cut point 25%, 30% and 35%, between WTS and SU under cut point 40% are statistically significant, respectively. In other cases, the differences are not significant. In addition, Cor and FS can always achieve best rankings except under cut points 40% while WTS achieves the best ranking under this cut point. This conclusion is quite different from that in [22] and [23], which indicate that the RF method is the most stable method. The reason is that we focus on the stability of feature selection methods for noisy SDD, while the studies [22], [23] aim at the stability of the methods to data perturbation for SDD.

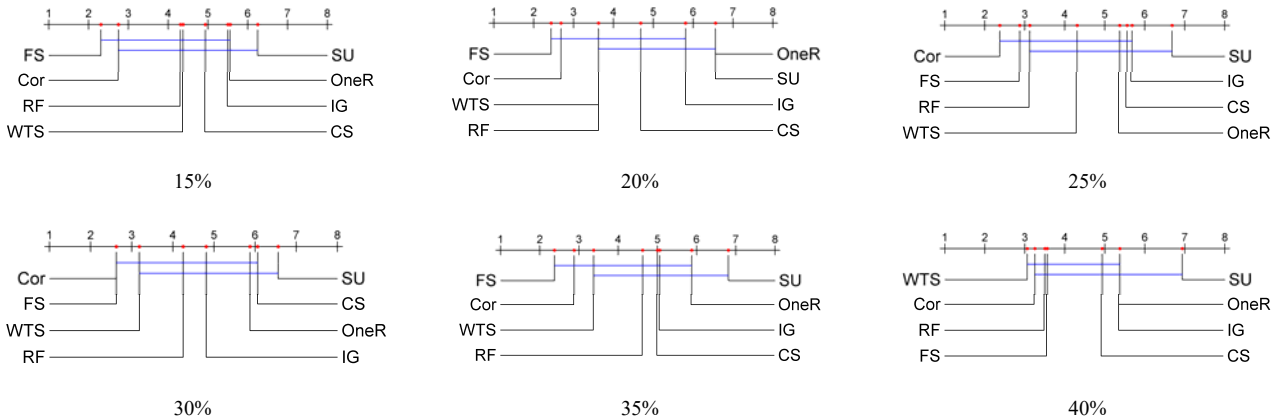


Fig.1. Comparison of all methods against each other in stability. Methods that are not significant different (the Nemenyi test, at p=0.05) are connected.

With the evidence provided by the above activities, we conclude that Cor and FS can achieve the most stable results for the noisy SDD on most cases.

VI. THREATS TO VALIDITY

For the generalization of our results, we carefully chose the NASA dataset which is commonly used in previous studies in software engineering domain [4], [17], [18], [19], [20]. Besides, previous work also conducted case studies on NASA dataset to investigate the effect of noise on SDD [8], [27]. So using NASA dataset can make our results more comparable and persuasive. For the bias in the choice of feature selection methods, as the first work to empirical study the equivalence and stability of feature selection methods for noisy SDD, we just select some typical methods for SDD. For the evaluation metrics, overlap index is reasonable to measure the extent of equivalence of feature selection methods since it has been used to evaluate the extent of equivalence of different machine learning techniques for defect prediction [24]. In addition, consistency index has been widely used to analyze the stability of feature selection methods for SDD in terms of data perturbations [15], [16], [31].

VII. CONCLUSION AND FUTURE WORK

This paper reports an empirical study of feature selection methods on SDD. This study involves two aspects: an equivalence analysis and a stability analysis. For the equivalence analysis, we raised the issue of the equivalence of different feature selection methods, that is, whether the methods have the similar effect to select relevant features. Analytic results by PCA and overlap index on eight projects of NASA dataset show that the studied methods are usually not equivalent to each other. The stability analysis with consistency index indicate that Cor and FS achieve better stability for noisy SDD.

In the future, we plan to take more feature selection methods, including feature subset selection methods, as our research objects. Meanwhile, we would explore the effect of different noise levels on the stability of feature selection methods for SDD.

ACKNOWLEDGEMENT

This work is partly supported by the grants of National Natural Science Foundation of China (No.61572374, No.U163620068, No.U1135005) and the Academic Team Building Plan from Wuhan University and National Science Foundation (NSF) (No. DGE-1522883).

REFERENCES

[1] T. Britton, L. Jeng, G. Carver, P. Cheak, T. Katzenellenbogen, Reversible Debugging Software. *Tech. Rep*., University of Cambridge, Judge Business School, 2013.

[2] C. Parnin, A. Orso, Are automated debugging techniques actually helping programmers? in: Proceedings of the 2011 *International Symposium on Software Testing and Analysis,(ISSTA)*, New York, NY, USA, pp.199–209, 2011.

[3] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu. A general software defect-proneness prediction framework. *IEEE Transactions on Software Engineering*, 37(3): 356-370, 2011.

[4] 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*, 34(4): 485-496, 2008.

[5] T. Menzies, J. Greenwald, and A. Frank. Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering*, 33(1): 2-13, 2007.

[6] M. A. Hall and G. Holmes. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data Engineering*, 1437-1447, 2003.

[7] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya. Choosing software metrics for defect prediction: an investigation on feature selection techniques. *Software: Practice and Experience*, 41(5): 579-606, 2011.

[8] C. Catal, O. Alan, and K. Balkan. Class noise detection based on software metrics and ROC curves. *Information Sciences*, 181(21): 4867-4877, 2011.

[9] S. Shivaji, E. J. Whitehead, R. Akella, and S. Kim. Reducing features to improve code change-based bug prediction. *IEEE Transactions on Software Engineering*, 39(4): 552-569, 2013.

[10] S. S. Rathore and A. Gupta. A comparative study of feature-ranking and feature-subset selection techniques for improved fault prediction. In *Proceedings of the 7th India Software Engineering Conference*. ACM, 7, 2014.

[11] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1): 63-90, 1993.

[12] A. K. Dubey and V. Yadava. Multi-objective optimization of Nd:YAG laser cutting of nickel-based superalloy sheet using orthogonal array with principal component analysis. *Optics and Lasers in Engineering*, 46(2):124-132, 2008.

[13] T. M. Khoshgoftaar, A. Fazelpour, H. Wang, and R. Wald. A survey of stability analysis of feature subset selection techniques. In *Proceedings of the 14th International Conference on Information Reuse and Integration (IRI)*. IEEE, 424-431, 2013.

[14] R. Real and J. M. Vargas. The probabilistic basis of the Consistency's index of similarity. *Systematic biology*, 45(3): 380-385, 1996.

[15] A. Kalousis, J. Prados, and M. Hilario. Stability of feature selection algorithms. *International Conference on Data Mining (ICDM)*, 218-225, 2005.

[16] S. Alelyani, Z. Zhao, and H. Liu. A dilemma in assessing stability of feature selection algorithms. In *Proceedings of the 13th International Conference on High Performance Computing and Communications (HPCC)*. 701-707, 2011.

[17] M. Shepperd, Q. Song, Z. Sun, and C. Mair. Data quality: Some comments on the NASA software defect datasets. *IEEE Transactions on Software Engineering*, 39(9): 1208-1215, 2013.

[18] Y. C. Liu, T. M. Khoshgoftaar, and N. Seliya. Evolutionary optimization of software quality modeling with multiple repositories. *IEEE Transactions on Software Engineering*, 36(6): 852-864, 2010.

[19] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto. Automated parameter optimization of classification techniques for defect prediction models. *ICSE*, 321-332, 2016.

[20] J. Nam and S. Kim. Heterogeneous defect prediction. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering*. 508-519, 2015.

[21] P. He, B. Li, X. Liu, J. Chen, and Y. Ma. An empirical study on software defect prediction with a simplified metric set. *Information and Software Technology*, 59: 170-190, 2015.

[22] H. Wang, T. M. Khoshgoftaar, and R. Wald. Measuring robustness of feature selection techniques on software engineering datasets. In *Proceedings of the 12th International Conference on Information Reuse and Integration (IRI)*. IEEE, 309-314, 2011.

[23] H. Wang, T. M. Khoshgoftaar, R. Wald, and A. Napolitano. A novel dataset-similarity-aware approach for evaluating stability of software metric selection techniques. In *Proceedings of the 13th International Conference on Information Reuse and Integration (IRI)*. IEEE, 1-8, 2012.

[24] A. Panichella, R. Oliveto, and A. De Lucia. Cross-project defect prediction models: L'union fait la force. In *Proceedings of the Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE)*, 164-173, 2014.

[25] T. M. Khoshgoftaar, K. Gao, and A. Napolitano. An empirical study of feature ranking techniques for software quality prediction. *International Journal of Software Engineering and Knowledge Engineering*, 22(02): 161-183, 2012.

[26] K. Gao, T. M. Khoshgoftaar, and A. Napolitano. Impact of data sampling on stability of feature selection for Software Defect Data. *In Proceedings of the 23rd International Conference on Tools with Artificial Intelligence*. IEEE, 1004-1011, 2011.

[27] J. Van Hulse and T. Khoshgoftaar. Knowledge discovery from imbalanced and noisy data. *Data and Knowledge Engineering*, 68(12): 1513-1542, 2009.

[28] Shivaji S. Efficient bug prediction and fix suggestions[J]. Dissertations & Theses - Gradworks, 2013.

[29] Z. Xu, J. Liu, Z. Yang, and X.Jia. The Impact of Feature Selection on Defect Prediction Performance: An Empirical Comparison. *International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 309-320, 2016.

[30] K. Gao, T. M. Khoshgoftaar, H. Wang, and N Seliya. Choosing software metrics for defect prediction: an investigation on feature selection techniques. *Software-practice & Experience*, 41(5):579-606, 2011.

[31] H. Wang, T. M. Khoshgoftaar, and R. Wald. A Study on First Order Statistics-Based Feature Selection Techniques on Software Metric Data. *SEKE*, 7-472, 2013.

[32] Z. Xu, J. Liu, and Z. Xia. Complete experimental results. [Online]. http://pan.baidu.com/s/1mic7ep6

A Membership-based Multi-dimension Hierarchical Deep Neural Network Approach for Fault Diagnosis

Liangliang Li
School of Computer Science and Technology
Tsinghua University
Beijing, China
Email: lll15@mails.tsinghua.edu.cn

Dai Guilan, Zhang Yong
Department of Computer Science and Technology and
Research Institute of Information Technology
Tsinghua University
Beijing, China
Email: {daigl, zhangyong05}@mail.tsinghua.edu.cn

***Abstract*—Accurate fault prognosis of machine component is important to maintain industry operation system. Faults analysis can be very helpful in fault early warning and reducing maintenance cost. The goal of our work is to design an integrated approach of machine faults analysis. A method widely used is Fuzzy Neural Networks (FNNs), but such method lacks of flexibility. We present a Membership-based Multi-dimension Hierarchical (MMH) neural network model to jointly include new feature selection approaches and generalized membership operators. MMH model is an adaptive model that employs modified KPCA and Back Propagation algorithm respectively. By introducing optimized KPCA we can extract features of higher importance that are appropriate for fault diagnosis. Our prediction model is inspired by the traditional fixed membership. In our approach, an observing value will be segmented into multiple dimensions where each dimension captures deep structural information in the network. The transformation is updated by back propagation. The proposed approach takes advantage of membership thinking and benefits from large learning capacity of deep neural networks (DNNs). This is aiming to take advantage of membership thinking and neural network deep learning abilities. Experimental results on public datasets demonstrate the superiority of our model that has the character of faster convergence, which also improving the accuracy by an average of 5% for fault prediction.**

***Index Terms*—Feature Selection; Modified KPCA; Back Propagation; Multi-dimension Hierarchical Neural Network**

I. Introduction

Growing attentions on resource shortages around the world have led to an increasing number of researches on improving the energy efficiency. At the same time, machine maintenance and repairs have played an indivisible role in energy consuming. It has been reported that faults in machine may increase about 15% of energy consumption [1], which may also result in many other additional costs.

Fault diagnosis and resolution in a system network are essential for clearing faults that manifest in an electrical sensor transmission or distribution network. Many studies have been carried out on the use of intelligent methods for fault diagnosis in an electrical system.

In the case of faults analysis, we usually have different dimensions of fault feature indication data which is represented as $x_1, x_2, \cdots, x_D$ and $F(x)$ that is on behalf of the fault type of comparison given a series of faults feature observations:

$$F(x) = f(\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_n})$$

assuming that the fault type ranges from f_1 to f_c, from this point, we hope to get implicit relationships from $\boldsymbol{x_n}$ to $F(x)$ in the real application scenario.

We get used to analyze and mine this set of D-dimensional vectors, but the complexity of many machine learning algorithms is closely related to the dimensionality of the data, so it is necessary for us to reduce the dimensionality of the data first.

Principal component analysis (PCA) [2] and kernel PCA [3] [4] are well-known methods in feature engineering. However it is still not enough for a KPCA algorithm solving faults feature selection. The existence of noise will keep on disturbing eigenvalues. In this paper, we propose a integrated algorithm combining the original eigenvectors' importance with the final faults type. Our main contributions are listed as follows:

1. By calculating the between and within class in a new way, it can minimize the impact of uncertain factors and increase the benifits of reducing dimensions of input features.
2. When processing the faults diagnosis problem, we find that the existing methods are lacking of mining information of each input dimension. Based on a membership algorithm, we change the structure of now existing multiple level proceptron by seperating input layer into several patchs, also we remove the full connected edge to the hidden layer in order to keep locality of each dimension feature contribution.

The remainder of the paper is organized as follows. Section II describes related work on fault diagnosis algorithm. In Section III, we present how to make fault feature extracted and evaluated by putting the faults type information into consideration before modeling. The new idea of MMH modeling for fault diagnosis problem is provided in Section IV. The experiment on public UCI datasets as well as discussions on baseline algorithm is shown in Section V. Finally we conclude our work in Section VI.

[1]DOI reference number: 10.18293/SEKE2017-074

II. Related Work

The study of fault diagnosis and prognosis recently have concentrated on theoretical research, mainly based on fuzzy theory, pattern recognition, bayes rules, logistics regression, neural network algorithms and so on. Others are focusing on building deep learning models to infer the relation between data and fault results or estimate the probability of faults occur. Below we highlight a few and explain what advantages and drawbacks they have.

- **Classical Fuzzy Set Interface Theory.** Previous research [5] has been done extensively concentrated on inference system design. Fuzzy rule based system (FRBS) deals with IF-THEN rules. FRBS consititute an extension to the classical fuzzy rule inference [6] [7].
- **Deep Learning Models.** Deep learning so long has became a point of focus as it is the skilled-expert in the domain of complex problems. In particular of fault diagnosis domain, different neural network (NN) models are proposed to fitting various background [8] [9] [10].
- **Hidden Markov Model.** A classification method [11] for reluctance motors' fault diagnosis using HHM is carryed out and shown that parameter learning need huge a mount of histrorical data.

III. Faults Feature Extraction

A. Principal component analysis (PCA)

Principal Component Analysis (PCA) analysis is an important means of dimension reduction. It applies a linear correlation transformation on original data features which can explain most of the datasets information in new scope.

Given a set of centered input vectors $\boldsymbol{x_t}$ $(t = 1, \cdots, n)$, and each of which is one of m dimension: $\boldsymbol{x_t} = (x_t(1), x_t(2), \cdots, x_t(m))^T$ then we have the input data matrix $X_{n\times m}$ (usually $n > m$), In general, we will select the eigenvector in which the largest eigenvalues are located. The information in these directions is rich, and is generally considered to contain more information of interest.

B. Theory of Kernel Principal Component Analysis (KPCA)

On account of there are some limitations of PCA, there is no way for the existence of high-order correlation, Kernel PCA can be introduced, using a kernel function we can transform the nonlinear correlation into a linear one. Given a set of input data $\Phi(x_i), i = 1, 2, \ldots, n$ for this discussion, the covariance matrix $\bar{C}$ of centralized data: $\bar{C} = \frac{1}{n}\sum_{i=1}^{n}\tilde{\Phi}(x_i)\tilde{\Phi}(x_i)^T$ Now finding the eigenvingue and eigenvector of $\bar{C}$ and kernel matrix is respectively donated as λ_c, $\tilde{\lambda}_k$ and υ_c, α_c

$$\bar{C}\upsilon_k = \lambda_k\upsilon_k(k = 1, 2, \cdots, D) \qquad (*) \tag{1}$$

the final conclusion by reducing both sides of the equation (1): $\lambda_c = \frac{\tilde{\lambda_k}}{N}$, $\alpha_c = \frac{1}{\sqrt{\tilde{\lambda_k}}}\tilde{\Phi}\alpha_k$. If the adoption of the kernel function is Radial Basis Function,

$$k(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2}) \tag{2}$$

here σ is the only parameter of function. In general, KPCA realize non-linear transformation between the data space and feature space through the kernel function.

C. Modified Kernel Principal Component Analysis

The traditional KPCA algorithm only considers the maximum information content of the reserved feature space and does not consider whether these information quantities are effective for classification.

Here, we reconsider the degree of dispersion among the intra-classes and inter-classes, which can both retain good dimension reduction and more conducive to the fault pattern classification.

Before we want to balance the degree of aggregation within class and between classes for each feature vector, a important notation firstly is introduced to represente one class center as $\bar{x}_i = \frac{1}{n_i}\sum_{p=1}^{n_i} x_{ip}, i = 1, 2, \cdots, c$, where c stands for the number of fault classes, n_i is the total number of labeled class i and x_i is the principal component after kernel transformation. The within class distance:

$$W_\sigma = \frac{1}{n_i}\sum_{i=1}^{c}\sum_{q=1}^{n_i}\|x_{iq} - \bar{x}_i\| \tag{3}$$

in the equation, we can calculate each W_σ vary from the extracted dimension d, also the inter-class discretization degree of each eigenvector is:

$B_\sigma = \sum_{i=1}^{c}\sum_{j=i+1}^{c}\|m_j - m_i\|$ where $m_i = \sum_{i=1}^{n_i} x_i$. If we get a bigger between class value B_σ and a smaller within class value W_σ, the more it is with the ability to distinguish categories. Intuitively, the definition of χ is $\chi = \frac{W_\sigma}{B_\sigma}$.

IV. Faults Diagnosis Modeling

After selecting the most informative feature in section III. In this part, we present the wide and multiple neural network and compare it with the traditional model we mentioned in related work. we want to highlight a few previous work by applying neural networks in the domain of fault diagnosis.

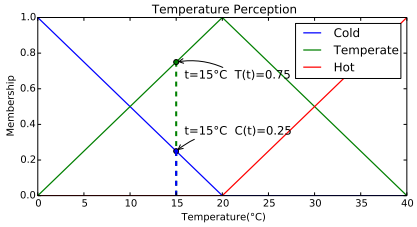


Fig. 1. Example of a tempreture degree membership.

Figure 1 shows a temperature perception in a real scene. In the theory, we assume that any temperature value will correspond to a linguistic description value when a fuzzy rule needs to be applied, or findind the corresponding exact value range. In the figure, when the temperature appears $15°C$, we subjectivity feel that the temperature value is a degree of cold is *0.25*, a moderate degree of *0.75* or when the description of the value of the cold, the corresponding temperature range of $0°C$ and $20°C$.

Here goes our Membership-based Multidimensions model assumptions as follows:

1. Each observed measurable value υ (normalized) will consists by multi-tuples $\mathcal{M}(\upsilon_1, \upsilon_2, \cdots, \upsilon_d)$, d is a multi-dimension parameter that represent the disperse level.
2. The summation of υ equals to a fixed setting: $\sum_{i=1}^{d} \upsilon_i = s$, s here represents the multi-dimension central degree. (e.g., specific $s = 1$, due to the result of normalization, it somehow play as a limitation to d-dimension tuples)
3. Such extended dimensions in a descriptive way (like Fig 1) are independent.
4. Nonlinear relations exist during the learning the faults classes patterns.

Consider the basic structure of a back-propagation network with a single hidden layer, as shown in Figure 2:

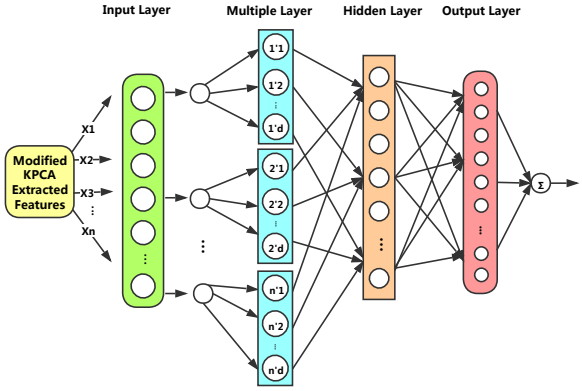


Fig. 2. Model of faults diagnosis based on multiple dimensions neural network.

We divide the input each of the n dimensions into independent description vectors 1-d of the d dimension. If we want our multiple layer outputs are scaled to $[a, b](a, b \in (0, 1])$, the sigmoid function is given as follows: $s(x_i) = \frac{1}{1+e^{-\alpha_i(x_i-\beta_i)}}$ where: $\alpha_i = \frac{(Q_i-1)\ln(\frac{b}{1-b})}{x_{i,max}-x_{i,min}}$, $\beta_i = \frac{Q_i x_{i,min}-x_{i,max}}{Q_i-1}$, $Q_i = \frac{\ln(\frac{1-b}{b})}{\ln(\frac{1-a}{a})}$, and $Q_i \neq 1$. Here α_i *and* β_i are parameters of the active function. The purpose of this approach is to avoid x_i spills out and can be mapped to $(0, a)$ or $(b, 1)$ respectively.

If we have N training examples and C classes label of fault diagnosis then the loss for our prediction $\hat{y}$ with respect to the true labels y is given by:

$$Loss(y, \hat{y}) = -\frac{1}{N}\sum_{n\in N}\sum_{i\in C} y_{n,i} log\hat{y}_{n,i} + l2\_reg\_para * (a_0 + a_1 + \cdots + a_i) \tag{4}$$

Before we doing the back propagation, the difference vector should be compute as : $\delta a = a_2 - y$, then we describe the backpropagation in a functional way, and specially the w_0 matrix update: $\delta a_0 = \delta z1 \cdot w_1^T$, $\delta z_0 = \delta a_0 \cdot S'(z_0)$, $\delta b_0 = \delta z_0$, $\delta w_0 = x_i^T \cdot \delta z_0$, $\sum_{i=1,d,2d,\cdots,nd}^{i+d-1} w_{0_i} = s$. By finding parameters that minimize the loss of our training data, variations such as SGD (stochastic gradient descent) or minibatch gradient descent typically perform better in practice.

$$w_{new} \leftarrow w_{old} - \eta \cdot \delta w_{old} \tag{5}$$

In the process of gradient descent, to prevent local shock, we also introduce decaying learning rate over time: $\eta = \eta_0 \cdot e^{-d_0 t}$.

V. Experiments

In this section, we empirically study the performance of Modified Kernel PCA and MMH model in public date sets.

Datasets: we used two large UCI datasets: *Secom* [12] and *Sensorless Drive Diagnosis* [13]. The first is a binary classification problem where data were taken from a semi-conductor manufacturing process and used to select most relevant signals. The second dataset was extracted from 11 different labels motor current with intact components.

TABLE I
DATESETS FOR KPCA AND MMH MODEL

Dateset	Dimens.	Classes	Instances	Train Prop.
Secom	591	2	1567	-
Sensorless	49	11	58509	0.8

A. Feature Extraction Task

We solve feature extraction task by compareing the PCA and KPCA with *RBF* kernel in the first step.

TABLE II
COMPARATION IN SVM RESULT BY ORIGINAL AND MODIFIED KPCA

Method	Reduced Dimensions	Accumulate Contribution	Accuracy
Original KPCA	6	89.20%	72.24%
	9	90.73%	79.92%
	12	96.21%	82.21%
Modified KPCA	6	90.82%	**85.27%**
	9	92.12%	**89.23%**
	12	97.44%	**92.27%**

In tabel II, we put the Original KPCA, Modified KPCA into the based SVM classificator. After 10-fold cross-validation, chaging the dimensions from 6, 9, 12, our proposed Modified KPCA algorithm can obtain more valuable information form each disparate dimension. It can more likely find the number of selected features with the strongest causal effect relationship.

Using Modified KPCA we proposed, comparing with PCA, KPCA in the same dataset experiment, we can find modified kpca performs well and quickly from the view of aggregate proportion of importance. Although the contribution of the first dimension, PCA is little higher than KPCA and modified KPCA, as dimensions extend bigger, the accumulative proportion line of KPCA goes higher than naive PCA.

B. Faults Event Diagnosis Task

We cast faults event diagnosis tasks upon the UCI sensorless dataset. In this task, basically, we set up multi layer perceptron serve as a contrast to our proposed model with 3 hidden layer of $[100, 300, 100]$ and the learning rate is 0.05. Also, we experiment a few traditional method as baseline.

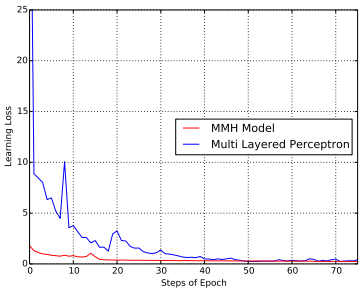


Fig. 3. Comparation of Loss Convergence Case

Since the existing open-source machine learning library can not meet the needs of the model, we build our own MMH model in Tensorflow. Under the premise of multiple dimensions equal to 3, the hidden layer structure is $[50, 300]$, and we set the adaptive learning rate η to 0.5.

To avoid overfitting, we introduced the normalization parameter of 0.005 to reduce overfitting probability as mentioned in the previous model introduction, and randomly discarded some of the neural during each iteration.The idea behind dropout is simple. The drop-out approach stochastically disables a fraction of its neurons. This can prevent neurons from co-adapting and forces them to learn individually useful features. The fraction of neurons we keep enabled is defined by the dropout probability with 0.1 input to our network.

In Figure 3 we can see a more undulating concussion loss convergence in common multi-layer perceptron. Our approach has more competitives in the aspect of convergence speed. Figure 4 show the diagnosis results we obtain through baseline algorithms and our MMH model. The results indicate that our method outperforms others by an average of 5% on the faults diagnosis problem.

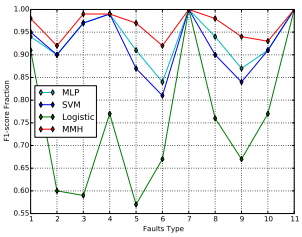


Fig. 4. Comparation of Different Algorithm's F1-score

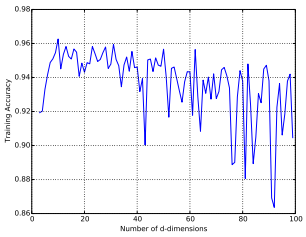


Fig. 5. Different d Dimension ValueS with Accuracy

In particular of Figue 5, when we take the range of d from 1 to 100, we find that the accuracy of the classification of MMH model increases first, and then slowly decline, basically meet the hypothesis, we speculate it is due to the fact that when d is becoming huger, the information is increased, and the contribution of this increment to the accuracy of the model is lower than the expected value of its own noise, which makes the accuracy of the model moves down. The certain range of evaluation is helpful for us to obtain the most appropriate d dimension mapping space according to the actual problem.

VI. Conclusion

The MMH neural network proposed in this paper is an effective and practical model for faults diagnosis and detection from a series of extracted features. MMH model gains the merit of classic neural network and benefits from valuable describing and multi-dimension information. When compared to traditional MLP model, experimental results shows that more latent information obtained by MMH are more convincing and quickly in convergence. Our Modified KPCA approach also provides a better supervised approach considering type labels which can accomplish more meaningful and useful feature extraction work.

Possible future directions for this work include limiting the summation of each multi-dimensions central degree with more certified principle and accelerating the process of Modified KPCA evaluating. As a result of calculating multi-dimensions vector enlarging the set size of multi-variables to estimate, our MMH model is more likely slower than a traditional MLP method. It would be of more importance to improve efficiency and reducing computational expense of vector extending for larger faults data input.

References

[1] D. Westphalen, K. W. Roth, and J. Brodrick, "System & component diagnostics," *ASHRAE journal*, vol. 45, no. 4, pp. 58–59, 2003.
[2] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.
[3] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.
[4] R. Rosipal, M. Girolami, L. J. Trejo, and A. Cichocki, "Kernel pca for feature extraction and de-noising in nonlinear regression," *Neural Computing & Applications*, vol. 10, no. 3, pp. 231–243, 2001.
[5] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoningi," *Information sciences*, vol. 8, no. 3, pp. 199–249, 1975.
[6] N. M. Pathade and S. M. Ansari, "Modeling, design and implement of fuzzy logic controlleron fpga robotic platform," 2015.
[7] K. D. Sharma, M. Ayyub, S. Saroha, and A. Faras, "Advanced controllers using fuzzy logic controller (flc) for performance improvement," *International Electrical Engineering Journal (IEEJ) vol*, vol. 5, pp. 1452–1458, 2014.
[8] C. P. Chen, Y.-J. Liu, and G.-X. Wen, "Fuzzy neural network-based adaptive control for a class of uncertain nonlinear stochastic systems," *IEEE Transactions on Cybernetics*, vol. 44, no. 5, pp. 583–593, 2014.
[9] Z. Gao, C. S. Chin, W. L. Woo, J. Jia, and W. Da Toh, "Genetic algorithm based back-propagation neural network approach for fault diagnosis in lithium-ion battery system," in *2015 6th International Conference on Power Electronics Systems and Applications (PESA)*. IEEE, 2015, pp. 1–6.
[10] M. D. Buhmann, "Radial basis functions," *Acta Numerica 2000*, vol. 9, pp. 1–38, 2000.
[11] B. Ilhem, B. Amar, and A. Lebaroud, "Classification method for faults diagnosis in reluctance motors using hidden markov models," in *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*. IEEE, 2014, pp. 984–991.
[12] https://archive.ics.uci.edu/ml/datasets/Dataset+for+Sensorless+Drive+Diagnosis.
[13] https://archive.ics.uci.edu/ml/datasets/SECOM.

Context Model Acquisition from Spoken Utterances

Sebastian Weigelt, Tobias Hey, Walter F. Tichy
Karlsruhe Institute of Technology
Institute for Program Structures and Data Organization
Karlsruhe, Germany
{weigelt|hey|tichy}@kit.edu

***Abstract*—Current systems with spoken language interfaces do not leverage contextual information. Therefore, they struggle with understanding speakers' intentions. We propose a system that creates a context model from user utterances to overcome this lack of information. It comprises eight types of contextual information organized in three layers: individual, conceptual, and hierarchical. We have implemented our approach as a part of the project PARSE. It aims at enabling laypersons to construct simple programs by dialog. Our implementation incrementally generates context including occurring entities and actions as well as their conceptualizations, state transitions, and other types of contextual information. Its analyses are knowledge- or rule-based (depending on the context type), but we make use of many well-known probabilistic NLP techniques. In a user study we have shown the feasibility of our approach, achieving F1 scores from 72% up to 98% depending on the type of contextual information. The context model enables us to resolve complex identity relations. However, quantifying this effect is subject to future work. Likewise, we plan to investigate whether our context model is useful for other language understanding tasks, e.g., anaphora resolution, topic analysis, or correction of automatic speech recognition errors.**

I. Introduction

The last decades have seen a growing interest in Spoken Language Interfaces (SLIs). The rise of intelligent assistants such as Siri, GoogleNow, and Cortana established SLIs in daily life [1]. One can easily book a restaurant, schedule a meeting, or make out whether one needs an umbrella by talking to a smart phone. However, such systems struggle with long utterances and complex relations. They are not always able to understand the speaker's intention. Humans naturally leverage information that embed the utterance into its context. Contextual information include statements expressed by the chosen words themselves as well as information related to the underlying concepts and the setting of the conversation. Humans exploit their general knowledge about grammar, concepts that exist in the world, and their perception to create an inner model that comprises several forms of contextual information. Without these, a set of instructions, such as "Close the fridge [...] Open the dishwasher [...] Then close all open appliances," could not be interpreted correctly. Current approaches cannot infer that the dishwasher should be closed because they are not able to answer the following questions: Apparently, 'all appliances' is a group of objects, but which objects are appliances? What does 'open' mean and which objects can be open? What can be inferred from the actions 'open' and 'close' in regard to the state of objects?

We propose a system that infers the required contextual information from the utterance automatically. First, we define eight types of context, organized in three layers of increasing abstraction. Then, we implement multiple techniques to generate a context model from spoken utterances: Our approach uses NLP techniques to cope with contextual information that can be extracted from the actual wording. We integrate knowledge databases, such as WordNet [2], to find relations between entities in an utterance. In addition, we use a domain model to extract contextual information concerning the environment of the conversation. Finally, our system infers further information by combining the different sources. This context model enables us to resolve complex relations as in the above example. We will discuss use cases in Section VII.

II. Related Work

Most spoken language interfaces do not incorporate explicit contextual information. However, many approaches model and use contextual information implicitly. Systems based on recurrent neural networks make use of the architecture of such networks [3]. Since 'old' information is constantly fed back into the network, current decisions are based on previous to some extent. Similarly, partially-observable markov decision process approaches model a kind of implicit contextual information in their belief systems [4]. Implicit context is also utilized by some knowledge-based systems. Active ontologies retain the information of previous activations in the network and are therefore able to use this information to interpret the current utterance [5]. Implicit knowledge is only accessible in the specific task and can neither be shared among different tasks nor addressed directly. Furthermore its impact is hard to quantify and evaluate.

Explicit contextual information is used by some approaches in natural language processing in robotics: The approach by Misra et al. uses situational contextual information to validate candidates for action grounding [6]. Fleischmann and Roy use the lexical context of actions to improve their approach to learn a mapping between spoken utterances and actions [7]. Another application is depicted by the approach of Bordes

DOI reference number: 10.18293/SEKE2017-083

TABLE I
OVERVIEW OF CONTEXT LAYERS AND TYPES

Layer	Type	Description
Individual	Entity	occurring things that can exist
	Spatial Deixis	spatial relations between entities
	Action	events occurring in context
	State Transition	state changes induced by actions
Conceptual	Concept	abstraction of entities and actions
	State	states individuals can be in
Hierarchical	Super Concept	hypernym relations between concepts
	Part-Of Relation	meronym relations between concepts

et al., which uses the identified concepts of preceding expressions to improve the language grounding of the current expression [8]. They all have in common that the contextual information primarily depicts the environment of the utterance. In contrast, we define a comprehensive context model that not only considers the environment but also the utterances itself, the expressed concepts and relations between them.

III. CONTEXT MODEL

The term *context* has several definitions [9], [10], [11]. Most definitions share the notion that context describes information that is used to understand the meaning of an artifact. The artifacts are parts of a communication situation, i.e., spoken utterances. The context of these artifacts includes information about the setting of the utterance, such as the place, time, the relation between the communicating partners and their mutual knowledge assumptions. In addition, afore-stated expressions in an utterance form a setting for the statement.

Since our approach is based on artifacts of spoken language, it is necessary to analyze which contextual information is retrievable from this limited input. The input consists of spoken utterances provided in a discourse. Sentences contain information about actions performed by subjects and treated objects. Additional expressions form relations between them, including spatial, temporal, or conditional. The subjects, objects, and actions are instances of concepts. For example, the utterance "The fridge is running," refers to the concepts `refrigerator` and `run_operate` (in contrast to `run_move`). In an utterance, many entities belong to the same concept. Similarly, many of the concepts are specializations of the same (or closely related) super concepts. Hence, concepts of instances from the utterance form a hierarchy of concepts. For instance the concepts `refrigerator` and `dishwasher` are manifestations of the super concept `white_goods`.

Based on these observations, we define three layers of contextual information that are extractable from spoken utterances: First the *Individual Layer* that deals with instances and the relations among them. Second, the *Conceptual Layer* that describes the concepts the instances form. And third, the *Hierarchical Layer* which considers generalizations of concepts. The layers constitute incremental layers of abstraction from the input. Table I summarizes the context layers as well as the types. We will discuss the layers and types in detail in the upcoming sections.

A. Individual Layer

The first layer of context describes the actual course of events. It concerns the actions, entities, etc. that occur in the spoken utterance. We define subjects and objects as acting and treated things and call this type *entity*. Taking this as a basis we augment relations between the entities: Locative relations, such as `cup:is_located_on:table` in the utterance "the cup on the table," are often used in spoken language. We denote this type of context as *spatial deixis*. Predicates determine the relation of entities. If a predicate expresses an action we call this information context of type *action*. For example, in "John, go to the table," the action is `go(who:John, where:table)`. The formalization of actions enables us to track which actions have been stated and which entities were affected by these. In some cases actions implicate a change of state of the treated entities; we call this kind of context *state transition*. For example, in the utterance "Open the fridge," the state of the entity `fridge` changes to `open`. State transitions provide valuable information to examine whether the utterance contains a valid sequence of actions. As we only consider spoken language, we have no information about the initial state of the entities. Thus the state transitions are the only way to obtain an image of the states the entities are in at distinct points in time.

B. Conceptual Layer

The *Conceptual Layer* contains the abstract concepts of entities, actions, and state transitions. To understand an utterance, humans shape concepts by leveraging the knowledge they have learned about the world and things that exist. This knowledge encompasses information about the entities and actions humans normally learn during their life, such as different meanings and synonyms of a word or relations between objects, e.g., that 'refrigerator' and 'fridge' refer to the same concept. Thus, we consider *concepts* as another context type. Note that whenever the same entity or action occur multiple times in utterances they shall refer to the same concept. Additionally, we define context of the type *state* as abstraction of state transitions: States are related to concepts, while state transitions are caused by actions. Thus, we see states as part of the *Conceptual Layer* rather than the *Individual Layer*. Note that different concepts can share the same state on the *Conceptual Layer*. E.g., the concepts `refrigerator` and `microwave_oven` might share the state `open`.

C. Hierarchical Layer

The *Hierarchical Layer* depicts super-conceptual and part-of relations of concepts. Humans do not stop the process of 'context shaping' with the concepts directly instanced in the utterance. In fact they use their knowledge to relate the concepts to each other. This happens in two ways. First, humans form superordinated concepts to perceive connections between concepts. We consider these relations as another context type named *super concept*. E.g. concepts such as `refrigerator` and `dishwasher` share the super concept

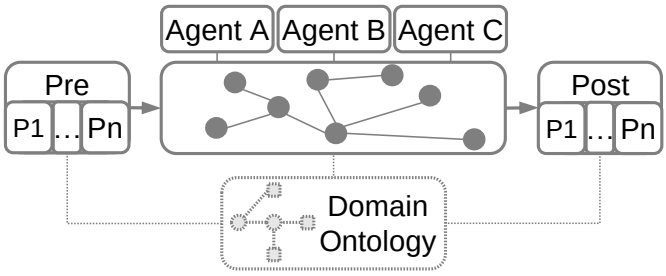


Fig. 1. Architecture of PARSE

TABLE II
DOMAIN ONTOLOGY STRUCTURE

Class	Description
System	Systems and sub-systems, i.e. API classes
Method	System functions, i.e. API methods
Parameter	Parameter names used by the system
Data Type	Data Types used by the system
Value	Value enumerations and ranges of data type values
Object	External objects, e.g., *a fridge* or *a cup*
State	States of the external objects, e.g., *opened* and *closed*

`white_goods` and therefore are concept-wise closely related to each other. Second, humans determine whether a concept depicts a part of another concept. We address this kind of relation with the context type *part-of relation*. E.g., in the utterance "Go to the fridge and open its door," the concept `door` is a part of the concept `refrigerator`.

IV. PARSE

Our work on context model acquisition from spoken utterances is part of the project *PARSE* [12]. The goal of the project is to enable laypersons to program in plain spoken English. Typical application areas of *PARSE* are robotics, home automation systems, and alike. To facilitate programming with spoken language the system must understand the user's intention. Thus, *PARSE* is actually a system for Spoken Language Understanding (SLU) [13]. To achieve deep SLU *PARSE* takes the approach of independent agents. Every agent is responsible for a certain SLU task. As SLU tasks are generally interdependent all agents work in parallel and therefore might benefit from results of each other. The strict separation of concerns additionally enables us to either build an agent knowledge-based or probabilistically according to the SLU task at hand and evaluate it intrinsically. The architecture of *PARSE*, which is illustrated in Figure 1, is separated in three independent parts: A pipeline for preprocessing, an agent-based main execution, and a pipeline for postprocessing. A graph serves as shared data structure for the agents. The preprocessing pipeline is meant for common natural language processing tasks, e.g., automatic speech recognition, shallow parsing, and named entity recognition. The user utterance is processed sequentially here; finally a basic graph for the main execution is built. The main execution is responsible for SLU. Agents for deep SLU transform the graph to publish their results. Hereby, a semantic representation of the input is created incrementally. SLU tasks encompass detection of actions, loops and conditions, context, topic and coreference analysis, etc. If the graph cannot be transformed to a proper intention model, the utterance is likely to be incomplete or ambiguous. In such situations the user is queried for clarification. The postprocessing pipeline maps the user's intention – modeled in the graph – to functions of the target system. Target systems are modeled in ontologies as proposed in our previous project *NLCI* [14]. We define a class hierarchy suitable for all target systems as shown in Table II. In [14] we have shown, that domain ontologies can be extracted semi-automatically from most APIs for end-user programming with small effort.

V. CONTEXT ACQUISITION

To generate the representation of contextual information described in Section III we use an incremental approach to construct the layers. We have implemented the context acquisition as an agent for *PARSE*. Thus, we can draw from information created by the preprocessing pipeline. This information includes parts of speech, lemmata, chunks, named entities, and semantic roles.

We start our analysis on the *Individual Layer*, the layer with the lowest degree of abstraction: We identify the described entities by analyzing the noun phrases. Additionally, we use a rule-based approach on parts of speech and chunks to add information such as the grammatical number, adjectives, quantifiers, and determiners. After extracting the entities we are able to search for spatial deixes describing relations between the identified entities. We accomplish this task by keyword matching on the expressions between the identified entities. This approach is reasonable, since in English grammar locative relations are usually expressed by prepositional phrases between nominal phrases they relate [15]. Finally, we extract the actions expressed in the utterance. To identify actions we combine a rule-based analysis of verb phrases with the information we get from the semantic role labeler SENNA [16] included in the preprocessing pipeline. Furthermore, we translate arguments of predicates to thematic roles from VerbNet [17] to relate actions and treated entities.

With the instances present, we are now able to infer the concepts and states on the *Conceptual Layer*. To generate concepts we first try to match entities with individuals from our domain ontology. We use the Jaro-Winkler distance [18] with a threshold of 0.92 and synonyms from WordNet to allow fuzzy matching. The threshold was determined empirically. If no proper match is found for an entity we query WordNet: First we use the full phrase that represents the entity. If unsuccessful we query WordNet again with sub-phrases of decreasing length (the head of the phrase is always included). The first match forms the concept. If there is no WordNet entry that matches the (sub-)phrase, no concept is created. For the following entities we first try to match the head of the phrase with an existing concept. Again we use the Jaro-Winkler distance and synonyms. If a matching concept is found we link the entity to the concept and proceed with the next entity. Otherwise the previously described generation process is invoked. For

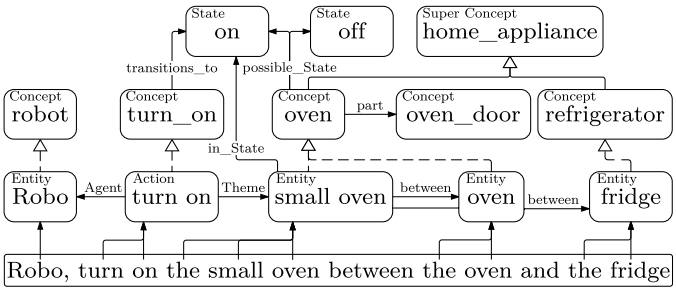


Fig. 2. Extract of the context model for an exemplary utterance.

all concepts we additionally retrieve synonyms from WordNet. This approach results in associating different entities, such as `white fridge` and `small fridge`, to the same concept `refrigerator`. For all created concepts we retrieve their possible states from the domain ontology and relate them to the concept. Then we examine whether the actions in the utterance cause any state transitions. We obtain this information from the domain ontology: E.g., the ontology contains the fact that the action `open` causes the state `closed` to change to `open`. For all state-changing actions we analyze the treated entities: We verify whether the entity's concept can be in this state. If so, we link the state to the entity. We allocate the same state to all following entities of the same concept until another state-changing action occurs. Then the analysis starts anew.

With the concepts at hand, we are able to form relations on the *Hierarchical Layer*. First, we use WordNet to build up a hierarchy of concepts: Apparently, the lowest common subsumer (LCS) of a pair of concepts in WordNet's hyperym hierarchy is a good candidate for a super concept. However, this naive approach would lead to many imprecise super concepts, such as `artifact` or `entity`. Therefore, we use the similarity metric defined by Wu & Palmer [19] with a threshold of 0.7 and a depth filter to avoid terms that are too generic. If one of the considered concepts is a hypernym of the other, the second becomes the super concept of the first. The hierarchy can be extended further: The created super concept can be used to find new super concepts. For example, if in the first iteration the super concepts `white_goods` and `kitchen_appliance` are created, the next iteration produces the super concept `home_appliance`. Second, we generate part-of relations: Some of these are included in our domain model and can be retrieved directly. If the ontology yields no information, we use meronyms from WordNet. These are solely used to create part-of relations among existing concepts to avoid false positives. However, parts retrieved from the domain ontology are added to the context model even if they had no concept representation before. The final context model consists of three layers with five context types denoted as nodes (entity, action, concept, super concept, and state) and three denoted as edges (spatial deixis, state transition, and part-of relation). Figure 2 shows an extract of the context model for the exemplary utterance "Robo, turn on the small oven between the oven and the fridge."

VI. Evaluation

To evaluate our approach to context acquisition we conducted a user study with 10 participants. We let all subjects describe tasks for a robot in two scenarios. We took recordings, transcribed them and provided a gold standard per context type. For each context type we calculate precision, recall and the F_1 score. In the upcoming subsections we first show the layout and implementation of our study, afterwards we discuss the results.

A. Experimental Design

Our user study comprises two scenarios containing a household robot in a kitchen setting. In both scenarios the robot should fulfill a certain task; in the first scenario it should fill a cup with water from the fridge, bring it to the user and afterwards put cups from the dishwasher into the cupboard. In the second scenario the robot is supposed to prepare an instant meal that is located in the fridge by putting it on a plate from the dishwasher and heat it in the microwave. Subjects are encouraged to describe the steps to accomplish the task to the robot. The instructions the participants received contained no concrete wording, just a high level description of the tasks and figures that showed the setting. We took continuous recordings, one per subject and scenario.

Ten subjects participated in the study, four female and six male. Nine were grad students from different departments and one was a computer science PhD student. All were native German speaker. However, all but one assessed their own English skills to be at least 'experienced' (*CEF* level C1). All participants gave descriptions for both scenarios. One participant repeated the recording for the second scenario because of an omission. Therefore we ended up with 21 recordings. As the subjects were able to describe the task as they liked, we received quite different recordings: They vary in length from 15 seconds up to 80 seconds and in instructions for the robot from a minimal set of 5 up to 22.

We transcribed all recordings according to the guideline by Kiesling et al. [20] and prepared a gold standard for each context type. We annotate the solutions as per-phrase labels. Thus, a phrase can have none, one, or more labels (depending on how many types of context it depicts). Annotating entities, actions and spatial deixes is straight forward. The gold standard for actions includes its argument relations. Thus, we can evaluate not only if the expected action is present but also whether all entities are identified correctly. We annotate states induced by the state transitions at the treated entities. This approach enables us to evaluate not only the state transitions but also the state inference mechanism: A recurring state annotated at different instances of an entity indicates that no state transition was identified. The expected states are those defined in the domain ontology. To assess the state generation on the *Conceptual Layer* we distinguish states and state connections: 'States' refers to the created states in the resp. scenario and 'state connections' to the created relations to concepts. We annotate concepts and super concepts as well as the possible states of the concepts at all phrases that can cause

TABLE III
EVALUATION: DATA OVERVIEW

	Scenario 1	Scenario 2	Total
Recordings	10	11	21
Words	734	811	1545
Phrases	467	543	1010
Instructions	121	143	264
Entities	199	233	432
Spatial Deixes	41	43	84
Actions	120	154	274
State Transitions	33	48	81
Concepts	274	320	594
States (Connections)	8 (306)	10 (338)	18 (644)
Super Concepts	88	56	144
Part-Of Relations	35	43	78

them. Hence, our evaluation is insensitive to consequential errors. Part-of relations are annotated at the phrase describing the part. As ground truth for the part-of relations we used the relations present in our domain ontology and meronyms from WordNet. For our scenarios, only a few relevant meronym relations can be found in WordNet. Thus, the majority of expected part-of relations are based on our domain ontology. Table III summarizes the recordings' meta data and shows the number of expected instances per context type. To retrieve the results of our agent we first run the preprocessing pipeline of *PARSE*. Afterwards we run our agent multiple times – until no further changes can be observed – without interference of other agents.

B. Results

To assess the soundness of our approach we calculate precision, recall and the F_1 score. Table IV summarizes the results of our evaluation. The results are promising. However, when we use more complex – and therefore error-prone – heuristics to generate context types the results diminish. Additionally, we generate results incrementally. Thus, failures occurring in the lower abstraction levels (or during preprocessing), such as incorrect entities or actions, cause failures in the higher levels. E.g., if a verb 'open' is falsely labeled as an adjective by the part-of-speech tagger this results in a missing action, a falsely identified entity (having open as describing adjective), and also causing a state transition expected not to be present because there is no action that triggers it. This kind of failure is propagated even further: Since we evaluate state transitions by examining the states of all occurrences of the same entity, a missing state transition results potentially in multiple incorrect 'current states' and therefore in false state transitions. Hence, the low recall of the context type state transition can partly be explained by aftereffects of the action analyses. Since the generation of actions is primarily based on the semantic role labeler SENNA, we expected to achieve F_1 scores around 75%[1]. Because our input examples are less complex than those from the CoNLL-2005 shared task, the generation of actions performs even better. However, our additional verb-phrase-based heuristic further improves recall

[1]SENNA achieves an F_1 score of 75,49% in the CoNLL shared task [16].

TABLE IV
EVALUATION: RESULTS BY CONTEXT TYPE

Context Type	Precision	Recall	F_1
Entity	0.972	0.975	0.973
Spatial Deixis	0.945	0.793	0.862
Action	0.852	0.762	0.804
State Transition	0.854	0.627	0.723
Concept	0.986	0.974	0.981
State	1.000	0.955	0.977
Super Concept	0.680	0.932	0.786
Part-Of Relation	0.897	0.959	0.927

by 5%. Nonetheless, missing or falsely created actions are unpleasant because they influence other context types. The relatively low recall of spatial deixes is caused by unexpected choice of words: The subjects used diverse phrases to express locative relations, some of which our approach is not capable to solve. For example, nested prepositional phrases, such as "put the meal from the fridge on the plate," may cause the locative relation to be ambiguous. This kind of ambiguity cannot easily be addressed by a keyword matching approach. We will investigate whether probabilistic methods are more suitable for such expressions. The generation of concepts from entities as well as the connection to the possible states is highly accurate; false positives and negatives result from failures in the preprocessing pipeline. Part-of relations are generated accurately; false positives are due to the use of WordNet: Since we do not resolve word ambiguities, we use all WordNet senses to find part-of relations. Sometimes this procedure causes failures, such as `plate` being a part of `table` instead of `dish`. The result of the evaluation of super concepts shows that our approach is focused on recall rather than precision. This objective is reasonable because we want to explore new context information and make it available to other language analyses. Wrong information can later be ignored, but information that is never generated remains unused forever. As with part-of relations, missing disambiguation of WordNet senses of the concepts is the reason of most of the false positives.

In summary our approach produces highly valuable results for the context types entity, spatial deixis, concept, state, and part-of relation. For the context types action and state transition an improved approach to semantic role labeling, i.e., adjusted to spoken utterances, would yield significantly better results. However, the evaluation shows that the current approach adds valuable context information anyway. The result of the context type super concept indicates that the approach is feasible but depends on correct WordNet senses.

VII. USE CASES

Our comprehensive context model is useful for resolving complex relations in spoken utterances. In this section we will show exemplary how we use the model in certain situations. For brevity we focus on identity relations.

First, we consider the sample utterance "Open the cupboard. Take the cup and close it." A naive approach would relate 'it' and 'the cup'. However, from the context model we know,

that the `cupboard` is open and can be closed, but a `cup` cannot. Therefore we can infer that 'it' more likely refers to 'the cupboard'.

The second example is, "There is a tumbler on the table. Take the glass and bring it to me." Here, the challenge is to understand that the expression 'the glass' is a generalization of the 'tumbler'. Since 'tumbler' and 'glass' are not synonymous, a simple WordNet query does not help. Neither a hypernym look-up in WordNet is helpful, as one cannot easily decide whether a hypernym is meaningful. Our model yields the information that `glass` is an appropriate super concept of `tumbler` and thus can be used as a substitute.

In the third example, "To prepare meringue take egg white, powdered sugar, and lemon extract. Put all the ingredients into the bowl," 'ingredients' refers to a group of previously mentioned entities. Our context model provides the information that `egg white`, `powdered sugar`, and `lemon extract` share the super concept `ingredient`. Together with the quantifier we are able to infer that 'all ingredients' refers to 'egg white', 'powdered sugar', and 'lemon extract'.

The last example, "Close the fridge [...] Open the dishwasher [...] Then close all open appliances," contains multiple challenges: First, one must resolve which are the 'appliances'. According to our context model `fridge` and the `dishwasher` share the super concept `appliance`. Second, the context model comprises the information that the fridge is closed and the dishwasher is open at the time 'all open appliances' is mentioned. Thus, we can infer that 'all open appliances' refers to the dishwasher.

VIII. Conclusion and Future Work

Acquiring a context model of spoken utterances is essential for understanding the speaker's intention. Without such information a spoken language interface could never fully interpret the relations in complex expressions. We have presented a new approach to generating a comprehensive context model from spoken utterances. Our approach generates eight context types, e.g., occurring entities and actions, the concepts of which they are instances, and relations between these as super and part concepts. The context types are organized in three layers representing different levels of abstraction.

The evaluation showed that the approach is promising: We achieve F_1 scores from 72% up to 98%, depending on the context type. However, the scores might be improved in several ways: Our generation of (super) concepts and part-of relations depends on the correct selection of the WordNet sense for a concept. Therefore, a comprehensive word sense disambiguation would improve our results. Moreover, our approach would benefit from a semantic role labeler adapted to spoken utterances. Since the semantic role labeler used by us is trained on textual input, it struggles with ungrammatical phrases common in spoken utterances. Consequentially, our generation of actions contains false positives. These are propagated to other context analyses, e.g., state transition.

However, the evaluation shows that the current approach adds valuable context information. We have shown that our context model can be used to resolve complex identity relations. The next step is to quantify this effect and extend this approach to coreference resolution in general.

Future work will focus on exploring further areas of application for our context model: We expect it to be useful for tasks such as topic extraction, time line analysis, correction of automatic speech recognition errors, and even word sense disambiguation. A partial context model might support word sense disambiguation, which in turn improves the context model.

References

[1] J. R. Bellegarda, "Spoken Language Understanding for Natural Interaction: The Siri Experience," in *Natural Interaction with Robots, Knowbots and Smartphones*, J. Mariani, S. Rosset, M. Garnier-Rizet, and L. Devillers, Eds. Springer New York, 2014.

[2] C. Fellbaum, *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[3] P. Xu and R. Sarikaya, "Contextual domain classification in spoken language understanding systems using recurrent neural network," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014.

[4] S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, "The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management," *Computer Speech & Language*, vol. 24, no. 2, Apr. 2010.

[5] D. Guzzoni, C. Baur, and A. Cheyer, "Modeling Human-Agent Interaction with Active Ontologies." in *AAAI Spring Symposium: Interaction Challenges for Intelligent Assistants*, 2007.

[6] D. K. Misra, K. Tao, P. Liang, and A. Saxena, "Environment-Driven Lexicon Induction for High-Level Instructions," *ACL (1)*, 2015.

[7] M. Fleischman and D. Roy, "Intentional Context in Situated Natural Language Learning," in *Proceedings of the Ninth Conference on Computational Natural Language Learning*, ser. CONLL '05. Stroudsburg, PA, USA: Association for Computational Linguistics, 2005.

[8] A. Bordes, N. Usunier, R. Collobert, and J. Weston, "Towards understanding situated natural language," in *International Conference on Artificial Intelligence and Statistics*, 2010.

[9] A. Duranti, *Rethinking Context: Language as an Interactive Phenomenon*. Cambridge University Press, May 1992.

[10] A. Fetzer, *Recontextualizing Context: Grammaticality Meets Appropriateness*. John Benjamins Publishing, 2004.

[11] R. Stalnaker, *Context and Content: Essays on Intentionality in Speech and Thought*. Oxford University Press, 1999.

[12] S. Weigelt and W. F. Tichy, "Poster: ProNat: An Agent-Based System Design for Programming in Spoken Natural Language," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE)*, vol. 2, May 2015.

[13] G. Tur and R. De Mori, *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. Wiley, Apr. 2011.

[14] M. Landhäußer, S. Weigelt, and W. F. Tichy, "NLCI: A Natural Language Command Interpreter," *Automated Software Engineering*, Aug. 2016.

[15] H. H. Clark, "Space, time, semantics, and the child," in *Cognitive Development and the Acquisition of Language*. Oxford, England: Academic Press, 1973.

[16] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (Almost) from Scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, 2011.

[17] K. K. Schuler, "VerbNet: A broad-coverage, comprehensive verb lexicon," *Dissertations available from ProQuest*, Jan. 2005.

[18] W. E. Winkler, "The State of Record Linkage and Current Research Problems," Statistical Research Division, U.S. Census Bureau, Tech. Rep., 1999.

[19] Z. Wu and M. Palmer, "Verbs Semantics and Lexical Selection," in *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*, ser. ACL '94. Stroudsburg, PA, USA: Association for Computational Linguistics, 1994.

[20] S. Kiesling, L. Dilley, and W. D. Raymond, "The variation in conversation (vic) project: Creation of the buckeye corpus of conversational speech," *Ohio State University, Columbus, OH*, 2006.

A Comparison of Two Model Transformation Frameworks for Multiple-viewed Software Requirements Acquisition

Bingyang Wei
Department of Computer Science
Midwestern State University
Wichita Fall, Texas, 76308
Email: bingyang.wei@mwsu.edu

***Abstract*—Multiple-viewed requirements modeling allows modelers to elicit the requirements of a system from different viewpoints. Requirements are then organized and encoded in different analysis models which collaboratively form an overall understanding of the system. Model transformations among those analysis models at this stage can be used as a way to acquire requirements knowledge, thus making the set of models complete and consistent. Two frameworks are found to support such requirements acquisition: the pairwise framework and the common representation framework. In practical applications, various factors need to be considered when requirements modelers choose between the two frameworks in order to acquire requirements by analysis model transformations. In this paper, we propose a set of criteria which provides a theoretical basis for comparing the two frameworks for their effectiveness of generating models and acquiring requirements in the context of multiple-viewed requirements modeling. The results of the comparison is then presented.**

I. Introduction

Properly eliciting and modeling the requirements are important to successful software development. Multiple-viewed requirements modeling is commonly used so that a system is observed from different viewpoints by different people, and the requirements are expressed in different types of analysis models. The creation of different types of analysis models for a system is parallel and iterative by nature and might be conducted by different requirements modelers. However, it is difficult for a modeler to know whether an analysis model is complete or what requirements are missing from the current version of a model [6]. The modelers need to be made aware of the missing requirements of analysis models. In order to make explicit the missing requirements in an analysis model, Delugach proposed the idea of conceptual feedback [4] which can provide prompts for the missing requirements to modelers (see Figure 1). This approach is based on the requirements knowledge overlap among analysis models of the same system. During requirements analysis stage, already-created analysis models ($Model_0$ to $Model_n$ in (b) of Figure 1) are transformed to a target analysis model $Model_x$. This process introduces new requirements to $Model_x$, which make it more complete, and more importantly, reveal some requirements acquisition opportunities to the modeler of $Model_x$.

DOI reference number: 10.18293/SEKE2017-070

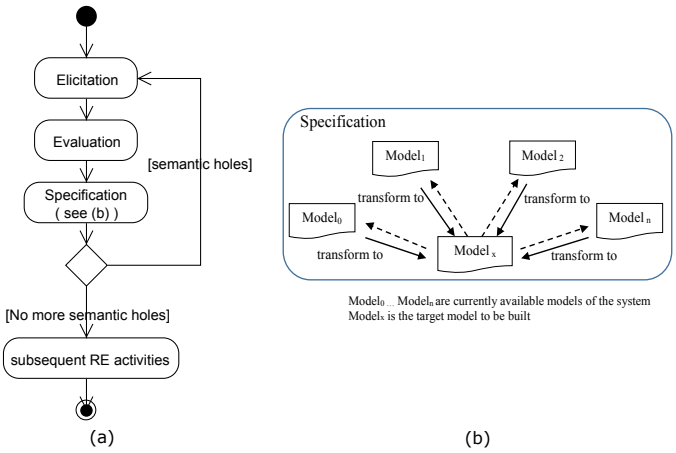


Fig. 1. Conceptual feedback in RE process

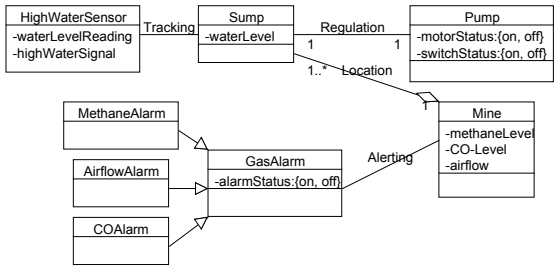


Fig. 2. Original class diagram before model transformations

A simple example of adopting the conceptual feedback approach to get more requirements and reveal requirements acquisition opportunities in a UML class diagram is shown in Figure 2 and Figure 3. Through model transformations, the original UML class diagram shown in Figure 2 is augmented by requirements from the current state diagrams and sequence diagrams of a Mine Safety Control system [10]. The resulting augmented class diagram is shown in Figure 3. Grey classes represent new classes that are added in through

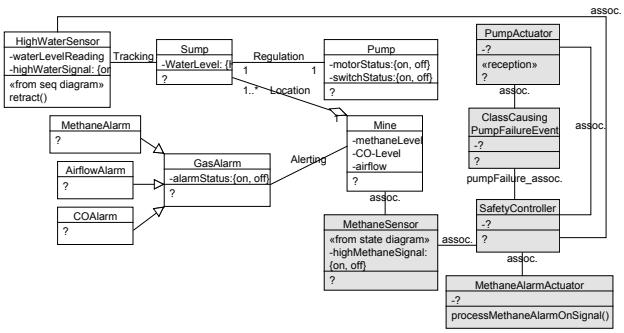


Fig. 3. Augmented class diagram after model transformations

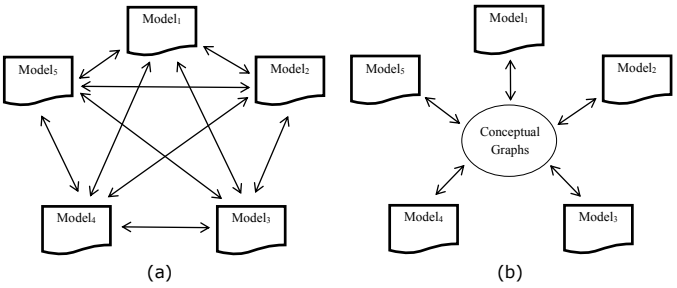


Fig. 4. PWF and CRF

model transformation. Note that the augmented class diagram contains question marks which denote the potentially missing requirements. In other words, they are requirements of the class diagram which cannot be automatically generated based on the existing requirements knowledge in other currently available models, therefore need to be explicitly provided by modelers. For example, class **`PumpActuator`** in Figure 3 has two question marks: an unspecified attribute and an unspecified operation.

Because of the presence of requirements acquisition opportunities, the augmented analysis model is considered as incomplete so modelers are invited to complete it by providing the missing requirements. This enables additional new requirements to be elicited by fixing those question marks (This process is shown as a feedback arrow from Specification to Elicitation in (a) of Figure 1). After that, the augmented model with question marks resolved would in turn affect other models (dotted arrows in (b) of Figure 1), causing further generation and completion processes in other analysis models. The process may repeat until no more new requirements knowledge can be acquired by transforming models, i.e., the set of models is internally complete and self-consistent.

There are two common used frameworks in Requirements Engineering community that can support software requirements acquisition by model transformations. They are the pairwise framework (PWF) and the common representation framework (CRF) shown in Figure 4. The former framework supports requirements acquisition by direct transformations between models whereas the latter by relying on a common knowledge representation that describes the semantics of the analysis models.

For PWF, transformations between each pair of UML diagrams have been studied by many researchers [5][7][8], and they collaboratively carried out a well-understood semantic description of model transformations in PWF. A summary of transformation rules between each pair of three UML diagrams is available in [11]. As for CRF, Wei and Delugach [13] proposed using conceptual graphs (CGs) [9] as the common representation in CRF to transform and acquire requirements for analysis models. A detailed description of model transformations with CGs in the CRF used for requirements acquisition is provided in their work [11][14][12].

An important research problem is that the two frameworks' effectiveness in transforming models and acquiring requirements knowledge is largely unknown. There is no theoretical basis for evaluating which framework is more effective in transforming models and acquiring requirements in the context of multiple-viewed requirements modeling. A detailed analysis of the capability of eliciting requirements from requirements modelers, the capability of preserving semantics and the extensibility of each framework, and an objective comparison is needed so that researchers and modelers can rely on this proposed criteria and comparison to choose between frameworks for addressing requirements acquisition problem.

The remainder of the paper is organized as follows: In Section 2, we explain the approach we use in order to compare the two frameworks for their effectiveness of exposing requirements acquisition opportunities and propose the evaluation criteria. In Section 3, the two frameworks are evaluated according to a set of criteria; the results of this comparison are presented. Section 4 discusses the limitations of this work and concludes the paper.

II. Comparison Methodology

In this section, details of the comparison are presented. These include the analysis models used in both frameworks, comparison procedures, software system examples and proposed criteria.

In this work, three types of UML diagrams are considered for both frameworks (Figure 5). They are class diagrams, state diagrams, and sequence diagrams which are among the most commonly used diagrams in UML for specifying an object-oriented system. The reason for choosing them is that the structure, state, and interaction views of a system provide a sufficiently broad range of the semantics of object-oriented models to show the generality of our approach. The conversions and comparisons are done by the author: conversion rules for both frameworks are derived based on the latest UML specification [1], and the conversions are conducted by strictly following the rules. We tried to limit the bias to the minimum, so it does not threat the validity of the results.

The way to conduct our comparison is shown in Table I. One model is considered as the target model, and two other models are transformed to it using one of the two frameworks: In PWF, two models are transformed to the third one directly, while in CRF, two models are first converted to CGs from which

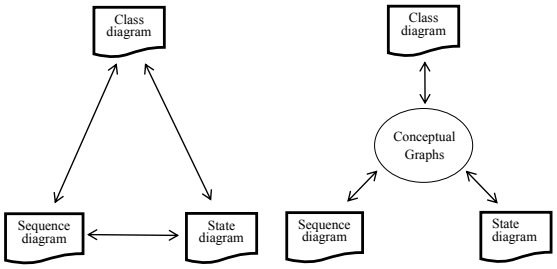


Fig. 5. Three UML diagrams in two frameworks

TABLE I
TRANSFORMATION STRATEGY

Transf. #	Source models	Target model
1	state diag., sequence diag.	class diag.
2	sequence diag., class diag.	state diag.
3	state diag., class diag.	sequence diag.

the third diagram is derived. The target models generated by different frameworks are then compared based on a set of criteria.

Three non-trivial software case studies are used. They are the University Information System (UnivSys.)[2], which is an information system; the Cryptanalysis System (Cryptanlys.)[3], which is an AI system, and the Mine Safety Control System (MineSys.)[10], which is a real time control system. The use of three case studies in three different domains makes sure that our comparison results are not dependent on the domain. The results of working out the three case studies in both frameworks are available in the appendices of [11].

A set of criteria that can be used to evaluate frameworks used for requirements knowledge acquisition is then proposed. Although only two specific frameworks are compared in this paper, these criteria are not limited to them. The criteria are meant to be applicable to any transformation framework that claims to address the requirements knowledge acquisition problem. The set of criteria is presented here:

1) Capability of acquiring missing requirements
2) Capability of generating definite requirements
3) Percentage of the missing requirements in generated UML diagrams
4) Extensibility
5) Knowledge acquisition effort

The reason these criteria are picked is the purpose of the model transformations in the two different frameworks: we are interested in the fact that which framework will produce more requirements acquisition opportunities for a modeler.

III. COMPARISON RESULTS

The results of the comparison are shown here. Each of criteria proposed in last section is explained in detail.

A. Capability of Acquiring Missing Requirements

Both model transformation frameworks can generate a target UML diagram and reveal requirements acquisition opportunities. In this work, a requirements acquisition opportunity refers to something that needs clarification in an augmented UML diagram. For example, a question mark on an association in a class diagram or an automatically generated class name like **`ClassCausingPumpFailureEvent`** in Figure 3. It reveals some missing requirement that a modeler needs to provide, and there is no way that a framework can generate that missing requirement automatically. Since the presence of requirements acquisition opportunities indicates the possible missing requirements knowledge, a high number of requirements acquisition opportunities in a UML diagram is a sign that more requirements knowledge will be potentially acquired from a modeler. This criterion evaluates the capability that a framework has to acquire the missing requirements. This is measured by counting the number of requirements acquisition opportunities in the UML diagrams generated by a framework. An advantage of using the quantity of requirements acquisition opportunities as the metric is that this only depends on the representation of UML diagrams and not on any subjective judgment of incompleteness or experience of requirements modelers. For example, different class diagram modelers will have different ways to complete the same augmented class diagram (Figure 3). So instead of asking several modelers to really fill in the holes and getting an average, we simply count the number of requirements acquisition opportunities that need to be filled, since these requirements (requirements acquisition opportunities) are missing for sure, and need to be provided by the requirements modelers.

During the comparison, given the same state and sequence diagrams of a software system, two class diagrams are generated by PWF and CRF, respectively; then the number of requirements acquisition opportunities yielded in each of the two generated class diagrams is counted. The types of requirements acquisition opportunities that we are counting in an augmented class diagram are listed under the gray row "In generated class diagram" in Table II. The same comparison process works for augmented state diagrams and augmented sequence diagrams. The types of requirements acquisition opportunities that we are counting in a generated state diagram are listed under the gray row "In generated state diagram" and the types of requirements acquisition opportunities that we are counting in a generated sequence diagram are listed under the gray row "In generated sequence diagram" in Table II. The complete results of evaluating the generated class, state and sequence diagrams for three case studies in the two frameworks are listed in Table II. Higher values are better. Based on the result, CRF generated more missing requirements than PWF did.

B. Capability of Generating Definite Requirements

In a generated target UML diagram, besides requirements acquisition opportunities, there are newly generated requirements that are certain. We named these requirements "definite inferred requirements" in this paper, and they do not need

TABLE II
COMPARISON OF THE CAPABILITY OF ACQUIRING MISSING REQUIREMENTS IN TWO FRAMEWORKS

	PWF			CRF		
requirements acquisition opportunities	UnivSys.	Cryptanlys.	MineSys.	UnivSys.	Cryptanlys.	MineSys.
In generated class diagram						
Number of unknown class names	7	8	4	7	8	6
Number of unknown attribute names	0	0	0	15	20	25
Number of unknown operation names	13	14	27	49	59	81
Number of unknown association names	12	11	13	12	11	15
Total	32	33	44	83	98	127
In generated state diagram						
Number of unknown/potential states	20	18	19	20	18	13
Number of unknown transitions	4	3	5	5	4	3
Number of unknown events	5	3	2	5	4	3
Number of unknown effects	0	0	0	7	7	5
Number of unknown guards	0	0	0	15	14	10
Number of unknown entry/exit, do activities	0	0	0	60	54	39
Number of state invariants	0	0	0	10	10	7
Total	29	24	26	122	112	80
In generated sequence diagram						
Number of unknown neighboring lifelines	6	8	8	6	7	8
Number of unknown messages	0	1	0	0	0	0
Number of unknown execution specs	0	0	0	25	37	58
Total	6	9	8	31	44	66

further clarification by modelers. For example, the method **`processMethaneAlarmOnSignal()`** in the augmented class diagram in Figure 3. This criterion evaluates the capability that a framework has to acquire the definite requirements knowledge. This is measured by counting the number of definite model elements in the UML diagrams generated by a framework. The types of definite requirements that we are counting in the three generated UML diagrams are listed in the first column in Table III. The results of evaluating the generated class, state and sequence diagrams for three case studies in two frameworks are listed in Table III. Higher values are better. Based on the result, the difference between PWF and CRF is small.

C. Percentage of the Missing Requirements in Generated UML Diagrams

This criterion evaluates the incompleteness of the UML diagrams generated by PWF and CRF. The incompleteness of a generated UML diagram is the number of requirements acquisition opportunities over the number of overall generated model elements in a generated UML diagram. The results of evaluating the incompleteness of generated class, state, and sequence diagrams for three case studies in two frameworks are listed in Table IV. For example, for the generated class diagram of the UnivSys. by CRF, it is 83.8% incomplete and 16.2% complete. Since the purpose of this paper is to evaluate requirements acquisition, higher percentages are better. Based on the result, CRF produces more incomplete diagram than PWF.

D. Extensibility

In this work, three types of UML diagrams are chosen for both frameworks. However, more UML diagrams can be added to expand the frameworks. This criterion evaluates the extensibility of a framework by measuring the amount of effort needed to introduce another type of UML diagram (activity diagram in this case) in both frameworks. For simplicity, only limited model elements in activity diagrams are considered: Activity partitions, Activity nodes, control flows, Forking, and joining. When a new type of model is introduced, new transformation rules need to be developed for both frameworks. We are counting the number of new rules developed for a framework to evaluate its extensibility. Lower values are better. In PWF, 26 new rules are needed to add a fourth diagram (the activity diagram), while in CRF, only 4 new rules are needed. Apparently, CRF is more extensible than PWF.

TABLE III
COMPARISON OF THE CAPABILITY OF GENERATING DEFINITE REQUIREMENTS IN TWO FRAMEWORKS

	PWF			CRF		
Definite requirements	UnivSys.	Cryptanlys.	MineSys.	UnivSys.	Cryptanlys.	MineSys.
In generated class diagram						
Number of definite classes acquired	5	4	8	5	4	8
Number of definite attributes acquired	2	3	11	1	2	5
Number of definite operations acquired	11	9	0	10	9	0
Number of definite association names acquired	0	0	0	0	0	0
Total	18	16	19	16	15	13
In generated state diagram						
Number of definite state machines	5	4	5	5	4	5
Number of definite states	0	0	0	0	0	0
Number of definite transitions	15	14	9	10	10	7
Number of definite events	10	8	7	10	10	7
Number of definite effects	9	9	7	9	9	5
Total	39	35	28	34	33	22
In generated sequence diagram						
Number of definite sequences	4	2	4	9	9	11
Number of definite messages	6	6	6	6	6	6
Number of definite execution specs	5	4	0	5	3	0
Number of definite states	6	10	14	6	10	14
Number of definite combined fragments	2	2	10	0	0	0
Total	23	24	34	26	28	31

TABLE IV
COMPARISON OF INCOMPLETENESS OF GENERATED UML DIAGRAMS BY TWO FRAMEWORKS

Generated	PWF			CRF		
UML diagrams	UnivSys.	Cryptanlys.	MineSys.	UnivSys.	Cryptanlys.	MineSys.
Class diagram	64.0%	42.7%	20.7%	83.8%	78.2%	54.4%
State diagram	67.4%	40.7%	27.3%	86.7%	77.2%	61.1%
Sequence diagram	69.8%	48.2%	19.1%	90.7%	78.4%	68.0%

E. Knowledge Acquisition Effort

This criterion is used to evaluate and compare the amount of effort needed to complete requirements acquisition opportunities in the UML diagrams generated by PWF and CRF. When a modeler is presented with UML diagrams with requirements acquisition opportunities, she needs to look at each hole and choose to either fill it in or delete it. So, in this work, the knowledge acquisition effort is measured by counting the number of requirements acquisition opportunities that needed to be considered (Table V). Higher values are better. Based on the result, CRF requires more effort than PWF.

IV. DISCUSSION OF THE RESULTS

Criterion 1 compared the capability of acquiring missing requirements of the two frameworks. Based on the results of applying both frameworks to three case studies, the CRF outnumbered the PWF in all generated UML diagrams in three case studies (see Table II). In other words, using CRF, more possible requirements may be acquired from a modeler in the multiple-viewed requirements modeling context. Currently, there is no metric to measure the usefulness of a requirements acquisition opportunity. Criterion 2 compared the capability of generating definite requirements in the two frameworks.

TABLE V
COMPARISON OF THE KNOWLEDGE ACQUISITION EFFORT NEEDED IN UML DIAGRAMS GENERATED BY TWO FRAMEWORKS

Generated	**PWF**			**CRF**		
UML diagrams	UnivSys.	Cryptanlys.	MineSys.	UnivSys.	Cryptanlys.	MineSys.
Class diagram	32	33	44	83	98	127
State diagram	29	24	26	122	112	80
Sequence diagram	6	9	8	31	44	66

Based on the results of applying both frameworks to three case studies, PWF outnumbered CRF in almost all generated UML diagrams in three case studies. In other words, using PWF, more determined requirements can be generated. This implies that PWF is better at generating target models from source models. However, the difference is not much. Given a set of requirements acquisition opportunities and a set of definite requirements in a generated target UML diagram, readers may wonder which one is more desired. In this work, we are aiming for knowledge acquisition, so more requirements acquisition opportunities are desired.

Built upon the previous two criteria, the comparison based on criterion 3 shows that UML diagrams generated in CRF are generally more incomplete than in PWF. This matches our first conclusion that CRF is more capable of exposing requirements acquisition opportunities in the multiple-viewed requirements modeling context.

Criterion 4 is crucial for framework developers since a framework grows when new types of diagrams are introduced. By introducing a new type of UML diagram, activity diagrams, in both frameworks, CRF is found to be clearly more extensible than PWF. This advantage becomes more evident when a framework supports more UML diagrams.

More requirements acquisition opportunities mean more effort involved in resolving them. In CRF, a modeler has to go through each requirements acquisition opportunity to decide if it is indeed useful and meaningful. In this work, we assume that every requirements acquisition opportunity takes the same amount of time to be completed. We conclude that it takes more time to complete a UML diagram generated by CRF than by PWF because more need to be taken into account.

Besides the purpose of requirements acquisition, CRF has the advantage of reasoning with the CGs Reservoir (the central conceptual graphs in Figure 5) which stores the entire requirements of the system. We recommend CRF as a better model transformation framework for the purpose of requirements acquisition.

Two current limitations are the lack of automation support and usage of simple case studies. Future work obviously will focus on automation and industrial examples. The current work assumes its input UML diagrams are syntactically and semantically correct. If errors exist, those errors will be converted to CGs or other UML diagrams and then used to infer incorrect or badly formed requirements knowledge. Retracting the CGs inferred from this wrong knowledge is a time-consuming and complex task that the current framework cannot handle.

V. CONCLUSION

In this paper, we proposed and conducted a comparison of two model transformation frameworks that can be used in requirements acquisition. Five general comparison criteria are provided for evaluating the frameworks so that modelers can reference them when choosing frameworks. Based on the results, common representation framework exceeds pairwise Framework in term of requirements acquisition.

REFERENCES

[1] OMG Unified Modeling Language™ (OMG UML), Version 2.5. object management group, 2015.

[2] S. W. Ambler. *The object primer: Agile model-driven development with UML 2.0*. Cambridge University Press, 2004.

[3] G. Booch, R. Maksimchuk, M. Engle, B. Young, J. Conallen, and K. Houston. *Object-oriented Analysis and Design with Applications, Third Edition*. Addison-Wesley Professional, third edition, 2007.

[4] H. S. Delugach. An approach to conceptual feedback in multiple viewed software requirements modeling. In *Joint proceedings of the second international software architecture workshop (ISAW-2) and international workshop on multiple perspectives in software development (Viewpoints' 96) on SIGSOFT'96 workshops*, pages 242–246. ACM, 1996.

[5] A. F. Egyed. *Heterogeneous view integration and its automation*. PhD thesis, University of Southern California, 2000.

[6] D. Firesmith. Are your requirements complete? *Journal of Object Technology*, 4(1):27–44, 2005.

[7] B. Nuseibeh, J. Kramer, and A. Finkelstein. A framework for expressing the relationships between multiple views in requirements specification. *Software Engineering, IEEE Transactions on*, 20(10):760–773, 1994.

[8] P. Selonen, K. Koskimies, and M. Sakkinen. Transformation between uml diagrams. *Journal of Database Management*, 14(3):37–55, 2003.

[9] J. F. Sowa. Conceptual structures: information processing in mind and machine. 1983.

[10] A. Van Lamsweerde et al. Requirements engineering: from system goals to uml models to software specifications. 2009.

[11] B. Wei. *A comparison of two frameworks for multiple-viewed software requirements acquisition*. PhD thesis, Ph. D. thesis, University of Alabama in Huntsville, 2015.

[12] 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.

[13] 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.

[14] 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.

Selection and prioritization of software requirements using the Verbal Decision Analysis paradigm

Paulo Alberto Melo Barbosa
University of Fortaleza - UNIFOR
Fortaleza – Brazil
albertobmap@hotmail.com

Plácido Rogério Pinheiro
University of Fortaleza - UNIFOR
Fortaleza – Brazil
placido@unifor.br

Francisca Raquel de Vasconcelos Silveira
University of Fortaleza - UNIFOR
Fortaleza – Brazil
raquel.vsilveira@hotmail.com

Marum Simão Filho
7 de Setembro College
Fortaleza – Brazil
marum@fa7.edu.br

***Abstract*— Selecting and prioritizing the most stable software requirements within a set of requirements and engaging them in releases that satisfy the most customers is a difficult task for the decision maker. Many methods have been employed to solve this type of problem. But we do not find many solutions that use Verbal Decision Analysis. Therefore, in this paper we aim to select and prioritize software requirements using Verbal Decision Analysis techniques as a tool, exploring the ZAPROS III-*i* method and comparing the results with those obtained by the NSGA-II, SPEA2 and Mocell metaheuristics and also with a random algorithm.**

Keywords- Software Release Planning, Multi-objective optimization, Verbal Decision Analysis, ZAPROS III-i

I. INTRODUCTION

Requirements are critical in the software development process. They provide the basis for estimating costs and effort, as well as allowing the development of estimates and test specifications [1]. During the software release planning process we can find several constraints, such as: project budget and precedence between requirements [2].

It is essential that the task of selecting and prioritizing requirements to take effect in the most efficient way possible. In the literature, we can find several methods to solve this problem. A common method is the use of metaheuristics that are algorithms used to solve optimization problems. Becceneri [3] say that metaheuristics is algorithmic tool, which can be applied to different optimization problems, with relatively small modifications, in order to make them adaptable to a specific problem.

Deciding which requirement to prioritize for implementation is typically a decision problem. We mean that a high degree of subjectivity is present in the decision-making process. This is a suitable scenario for Verbal Decision Analysis (VDA), which consists of an approach based on multi-criteria problem solving through its qualitative analysis [4], that is, VDA methods take into account criterion subjectivity.

Therefore, structured context in this work was the Software Releases Planning, which indicates a methodology that uses Verbal Decision Analysis to be used by software managers as a means to obtain an effective planning taking into account the selection of more priority requirements, having as criterion Stability of Requirements.

Our goal is to obtain a solution that contains an order of requirements to be implemented considering constraints (technical precedence between requirements and resources available to the project) and objectives (maximizing customer satisfaction by selecting the most important requirements for key customers and maximize stability among requirements, initially implementing those with the highest degree of stability). The results of the approach proposed here are compared with Barbosa's [5], which is an extension of Brasil's [16] work that used a similar methodology to solve problems with uncertainty of the number of requirements. In section 2 we will see works related to the planning of software releases and the approach proposed by [5]. In section 3 we will see the methods adopted to solve the problem proposed in this paper. In section 4 the results achieved and discussions and in section 5 the conclusions and proposals of future work.

II. RELATED WORK

The work published by Bagnall [6] deals with the determination of the requirements that must be executed for the next release of the software. The author predicts that customers have different levels of importance to the company and points out the requirements that have prerequisites. The algorithms applied in this strategy show the obtaining of quick solutions to small problems.

Therefore Greer et al. [7] say that defining which release to deliver the requirement is a decision that depends on several variables that relate in complex ways. The different perspectives of the stakeholders and the planning of releases, including effort restraint, are discussed.

Brasil [16] presents a structured approach in multi-objective optimization using metaheuristics for the problem of selection and prioritization of software requirements, considering the

DOI reference number: 10.18293/SEKE2017-150

inherent characteristics of real project and an uncertain number of requirements. Among these characteristics, she considered: i) stability of the requirement; ii) costs to implement the requirement; iii) technical precedence between requirements; iv) importance of stakeholders to the company and v) its preferences regarding requirements. The objective of this work was to compare the efficacy of metaheuristics in problem solving through performance measures and to compare the performance of metaheuristics with the result of the implementation and execution of a solution generated from a random algorithm.

The work differential [5, 16] was used as the selection criterion. Unstable requirements to be implemented late to avoid diverting project resources. The work of [5, 16] compared as solutions generated from the execution of the metaheuristics NSGA-II [8], Mocel [1] and a random algorithm. The results showed that metaheuristics were the best methods to reach higher quality solutions.

The task of ordering requirements can be complex and challenging. Many proposals for metaheuristic solutions are found in the research work. However, methods structured in VDA are little known in the field. In the opinion of [5, 6, 7, 16], the methodology of multicriteria support to the decision has several methods that can be applied in the most diverse problems. Therefore, the very choice of a multicriteria decision support method alone is already a multicriteria problem [9].

This work proposes to select and prioritize software requirements in the order in which they will be implemented using a VDA method known as ZAPROS III-*i* [10]. The results will be compared to those obtained when using quantitative methods (metaheuristics).

For the problem considered in this work, the application of the ZAPROS III-*i* method came from the acceptance of the method by the decision maker, which meant that the issues that were being presented to the decision maker made sense to him, and he was confident to answer them. In addition to this point, the need to evaluate the acceptance of the data, its properties used by the method, and whether the result supported in the decision-making process was exalted.

For the development of this work, we divided it into three stages: i) the generation of instances that represent a set of requirements to be implemented, ii) the classification of these requirements using ZAPROS III-*I*, iii) the comparison of the results obtained relating these with results obtained through the application of metaheuristic NSGA-II used in [5], SPEA2 [11], Mocell used in [1] and the solution of random algorithm and iv) Finally, we will evaluate the results. The figure 1 illustrates this flow.

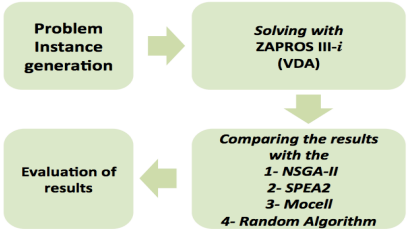


Figure 1. Considered Workflow

III. METHODS

The choice of a multicriteria method among those available, applied to a given context, should be adequate for the characteristics of the problem in question. An important point will be an evaluation of the problem, of the decision objects and the available information. The choice of method should be the result of an evaluation of the chosen criteria, the type and precision of the data, the form of the decision maker's thinking and his knowledge of the problem [12]. It is also emphasized that the direct consequence of possibility of choice between several methods that results can be discordant and even contradictory..

A. Intance Generation

Initially an application was developed to generate simulations of problems inherent in the software development process. In the work of [18] the results of the ZAPROS III-*i* method were compared with those obtained by the NSGA-II metaheuristic and good results were obtained with simulations involving 10 requirements and 5 clients. In this work we will increase the difficulty, especially the number of requirements that has been increased to 20, to analyze the behavior of the ZAPROS III-*i* method in this new situation involving a larger set of variables. Each simulation of the problem here contains:

i) Number of requirements to be implemented, which in this work were fixed in 20 requirements

ii) Number of customers interested in the project, which in this work was set at 7

iii) Cost Total of the project that in the experiments was considered between 70% or 80% of the total value needed to implement all the requirements

iv) Importance of each client to the company (whose value 01 represents the least important customer and 10 Importance),

v) Cost of each requirement (where 10 is the lowest cost and 20 the highest),

vi) Stability level of requirements, ranging from 1 to 10, where 1 means little stable and 10 is very stable

vii) Technical precedence matrix among the requirements, signaling that a particular requirement should only be implemented after another requirement indicated in that matrix

viii) Importance of the requirements for the customer that scores from 1 (minimum) to 10 (maximum) a customer's preference for a given requirement. The variations of these situations are presented in a simplified way in Table 1, which shows the four main characteristics adopted by this work

TABLE I. CHARACTERISTICS OF PROBLEM SIMULATIONS

File name	Number of require-ments	Number of Clients	Percentage of technical precedence for the requirement	Budget available for the project
I.20.7.10.70	20	7	10%	70%
I.20.7.10.80	20	7	10%	80%
I.20.7.20.70	20	7	20%	70%
I.20.7.20.80	20	7	20%	80%

B. Classification using ZAPROS III-i

When a decision can generate a considerable impact, such as management decisions, and must take into account some factors, the use of methodologies to support the decision making process is suggested, because choosing the inappropriate alternative can lead to waste of resources, time, and money, affecting the company [13].

The ZAPROS III method [14] is an evolution of the ZAPROS-LM one, with the application of the same procedure to elicit the preferences, but with modifications that make it more efficient and more accurate with respect to inconsistencies. Another difference between these methods is that the ZAPROS III is based on the elicitation of preferences around values that represent the distances between the evaluations of two criteria, called Quality Variations (QV), instead of comparing criteria estimates, as in its older version..

ZAPROS III-*i* method applies i) the Formal Index of Quality (FIQ) [14], which was used with the purpose of reducing the number of pairs of alternatives to be compared, ii) the ideas of comparison between alternatives through ordering the values of their quality vectors in ascending order [17] and iii) the comparison considering all possible alternatives for the problem, which can be used for solving complex decision making process.

Therefore, ZAPROS III-*i* presents a valuable alternative to solve requirements selection problems, since the opinion of the decision-maker is taken into account in this process.

C. Application of the methodology

To rank order the factors that project managers should consider when allocating tasks in distributed software development projects [17], we applied a methodology consisting of four main steps, which are explained next: 1) Identification of the Alternatives; 2) Definition of the Criteria and the Criteria Values; 3) Characterization of the Alternatives; and 4) The ZAPROS III-*i* Method Application.

1. The alternatives considered in this work were the 20 software requirements to be implemented. Each requirement has its own characteristics and will be known ahead.

TABLE II. CHARACTERISTICS OF PROBLEM SIMULATIONS

Criteria	Criteria values
A Cost	**A1** Requirement has low cost **A2** Cost of requirement is reasonable **A3** Cost of requirement is high
B Stability	**B1** The requirement will hardly change **B2** The requirement may change **B3** The requirement will change
C Stakeholders	**C1** The stakeholder is very important **C2** The stakeholder has partial and isolated importance **C3** The stakeholder is of little importance
D Customer requirement value	**D1** The requirement is of great value to the customer **D2** The requirement is of low importance to the customer

2. For the purpose of comparison, the criteria adopted in this study were the same as those adopted by [5] and [18]. As [3, 5] used quantitative methods and this work adopted the qualitative methodology, the data were converted to the qualitative methodology, where, for example, the Cost criterion represented on a scale of 10 to 20 was discretized into three values, whose values between 10 and 13.3 are represented by criterion A1, values between 13.4 and 16.6 represented by criterion A2 and values between 16.7 and 20 represented by criterion A3. This same methodology was adopted for the other criteria. Thus, the criteria were ranked from the most preferable (A1B1C1D1) to the least preferred (A3B3C3D2), according to table 2.
3. The characterization of alternatives was done according to the values contained in the requirements of the work problems of [5, 18]. Thus, considering the definition of criterion presented in the previous item and taking into account the criterion Cost, a requirement that presents Cost 15 in [3, 5] was classified as Cost A2 considering table 2. In the particular case, as the criteria 'Stakeholders' And 'Customer requirement value' have more than one customer by punctuating the same requirement, the arithmetic mean between the scores indicated by those clients for the classification represented in table 2 was considered.
4. After defining and characterizing the alternatives, we moved on to the stage of ordering. At this stage, we applied the ZAPROS III-*i* method to put in order the influencing factors, such that it is possible to establish a ranking of them, how make in [13].

In order to facilitate the decision-making process and perform it consistently, we used the ARANAÚ tool, presented in [10, 19, 20] which was implemented in Java platform, was first developed in [21] to support ZAPROS III method. In work of [13], was used the updated version to ZAPROS III-*i* method. The use of ZAPROS III-*i* method in the ARANAÚ tool requires four steps, as follows: 1. Criteria and criteria values definitions, 2. Preferences elicitation, 3. Alternatives definition, and 4. Results generation. The process runs as follows. First, we introduced the criteria presented in table 2 into the ARANAÚ tool. Next, the decision-maker decides the preferences. The interface for elicitation of preferences presents questionings that can be easily answered to obtain the scale of preferences. The questions provided require a comparison considering the two reference situations [4]. Once the scale of preferences is structured, the next step is to define the problem's alternatives. The quantitative values of [5] were used and converted to qualitative values.

D. Overlap of results

For each simulation of the files of table 1 of the problem two solutions were extracted: a set of solutions of the NSGA-II, SPEA2, Mocell, Random Algorithm execution and a solution of the ZAPROS III-*i*. NSGA-II and SPEA2 are similar and proposes a set of multi-criteria solutions coming from the Pareto front and the project manager selects the solution closest to his preferences. Mocell is a multiobjective algorithm based on the Genetic Algorithm model. The ZAPROS III-*i* methodology contains a single solution that was generated from the preferences indicated by the project manager. The random algorithm uses no technique and was used to depict an execution without any search strategy.

IV. RESULTS

At the end of each execution, the ARANAÚ [21] tool provided a ranking with the requirements ordered according to their execution priority. In table 3, we can see the result of this execution for file I.20.7.10.80.

TABLE III. RANKING OF ARANAÚ TOOL FOR THE FILE I.20.7.10.80

Ranking	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Requirements	8	10	19	20	7	12	15	18	5	14
Ranking	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Requirements	2	17	3	4	11	16	1	13	6	9

To facilitate comprehension and analysis, the results obtained were plotted and compared graphically. In Figure 2, 3, 4 and 5 we present the results of these executions with the NSGA-II, SPEA2, Mocell, Random Algorithm and the ZAPROS III-*i*, where each graph represents a simulation of the problem with its solutions.

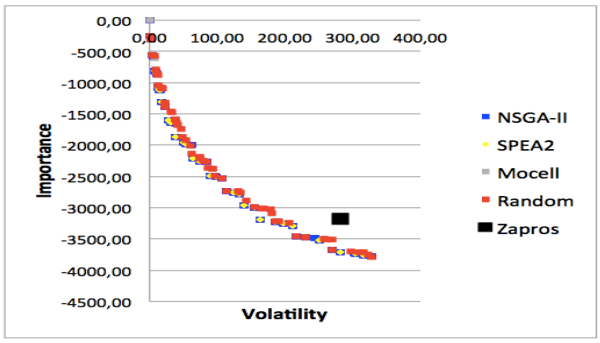


Figure 2. Result of file I.20.7.10.70

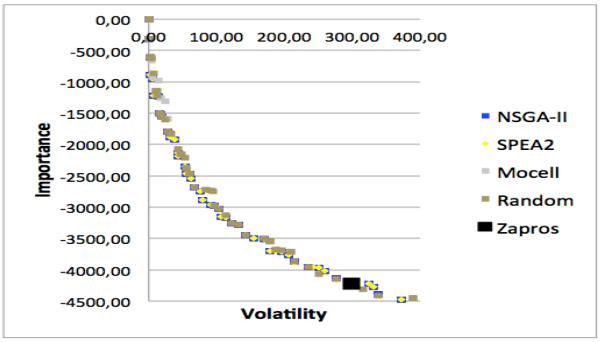


Figure 3. Result of file I.20.7.10.80

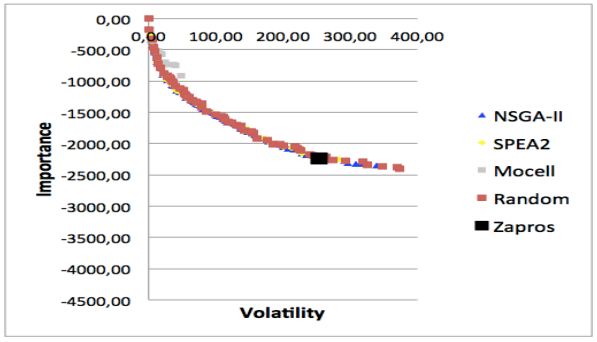


Figure 4. Result of file I.20.7.20.70

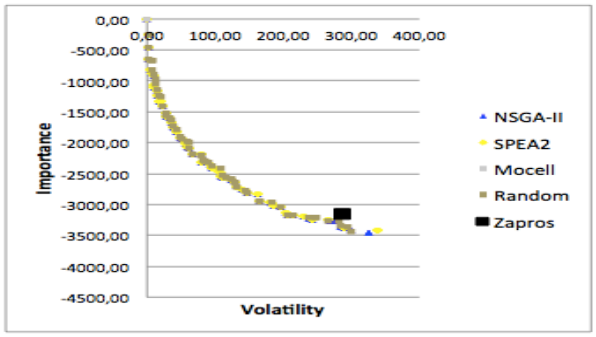


Figure 5. Result of file I.20.7.20.80

We can see five fronts of solutions. The Pareto fronts show a set points where each of them represent a possible ordering solution generated by each method. According to the legend of the graphs above, the colored dots correspond to the results obtained by the NSGA-II, SPEA2, Mocell, and Random. The black dots represent the solutions generated by ZAPROS III-*i*. Figures 2, 3, 4 and 5 represent solutions to problems with 20 requirements and 7 clients interested in these requirements. Figures 2 and 3 represent problems with 10% technical precedence between the requirements and figures 4 and 5, 20%. Accordingly, Figures 2 and 4 represent 70% of the budget available for the execution of the project, while Figures 3 and 5 represent 80%.

The stability criterion represented on the graph by the antonym (Volatility) was introduced in this work to assist the decision maker in choosing those more stable requirements to be implemented first. However, the solutions showed that the decision maker, in the tests applied, chose to value the importance that the important customer to the company gave to a given requirement. For this reason we see the points of ZAPROS III-*i* below the chart (valuing the client) and more right prioritizing more volatile requirements. This fact occurred in the four situations shown in Figure 2, 3, 4 and 5.

In Figure 2 and 3 we can see that the results obtained by NSGA-II and SPEA2 are superimposed. This fact occurs because of the similarity of the resolutive form chosen by these methods to solve the same problem.

Independent of the different situations, the solution generated by the ZAPROS III-*i* qualitative methodology was close to the solutions generated by the NSGA-II quantitative methods. This is a plausible fact if we consider the difference between the quantitative methods, adopted in NSGA-II, SPEA2, Mocell, and the qualitative methods, existing in ZAPROS III-*i*. This result shows that it is possible for qualitative methods to explore areas other than those already studied. The fact that a ZAPROS III-*i* solution, a decision maker's opinion, is close to solutions with complex search strategies for better solutions, shows that qualitative methods have a potential to solve this type of problem as well as metaheuristics They've been doing it for a while. In this case, the computational cost of the quantitative methods is replaced by the form presented to the decision maker, through the ARANAÚ tool that, as said, deals with the chosen alternatives and defines an order using the ZAPROS III-*i* method.

The solutions present in front of pareto can be used by a decision maker to facilitate the choice of the solution that is most appropriate for the proposed problem. The solutions found by the NSGA-II algorithm illustrate this Pareto front in Figures 3 and 4. As seen, the decision maker has a set of solutions to choose the one closest to his reality. The results obtained with the ZAPROS III-i qualitative method show very promising, since its solution was very close to the Pareto front, generated by metaeuristics that are quantitative algorithms. As in this case this solution was generated from information provided by the decision maker in a qualitative way, we conclude that this solution is what he really expected, since it is structured in real information provided by himself. In figure 3 and 4 the solution generated was equal to one of the solutions found by NSGA-II.

V. Conclusion and Future Works

The main contribution of this work is to apply a qualitative methodology structured in ZAPROS III-*i* Verbal Decision Analysis method to order software requirements and compare the solution generated with quantitative methodologies already known for doing this sort of ordering. The ARANAÚ tool provided support for this work, allowing good performance during testing and execution. As mentioned earlier, there are very few jobs that attempt to sort requirements using VDA methods. This work aimed to show that, although the results are not better than those obtained by the metaeuristics, the results obtained by ZAPROS III-*i* were very close to the front of the generated metaeuristic method.

In relation to the work developed by Barbosa [5], Brasil [16], and others to which these were extended, what was proposed in this work is a new methodology to prioritize software requirements using VDA giving a new alternative to the decision maker, which until then had only automated methods (metaheuristics).

As future work, we can increase the number of requirements to be sorted within the possibilities of VDA. This is a challenge because it is known that the problems solved by qualitative methods are limited in size due to the complexity of the methods themselves. To compare the results obtained by ZAPROS III-*i* with other metaheuristics that have populations of larger and more complex solutions. Increase the number of criteria to cover other types of problems related to ordering requirements. We can explore real-world problems by adapting the solutions according to the reality experienced by software development companies. We can apply other characteristics of the requirements, such as risk, estimated time for implementation and degree of complexity of the requirement.

Acknowledgment

The author Plácido Pinheiro is thankful to the National Counsel of Technological and Scientific Development (CNPq) for the support received for this project.

References

[1] DURILLO, J. J., NEBRO, A. J., LUNA, F., DORRONSORO, B., ALBA, E.: jMetal: a Java Framework for Developing Multi-Objective Optimization Metaheuristics. TECH-REPORT ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computación, Campus de Teatinos, Universidad of Málaga, Malaga (2006)

[2] RUHE, G., SALIU, M. O.: The Art and Science of Software Release Planning. IEEE Software, 47-53p (2005)

[3] BECCENERI, J. C.: Metaheurísticas e otimização. r.t. lac, inpe. Computers and Operations Research (2007)

[4] LARICHEV, O. I., Moshkovich, H. M.: Verbal Decision Analysis for Unstructured Problems, Boston: Kluwer Academic Publishers (1997)

[5] BARBOSA, P. A. M.: An Optimal-Based Approach to the Priorization of Software Re-quirements, Considering the Stability of the Requirement. Master Thesis – Academic Mas-ter in Computer Science, Ceará State University (2013)

[6] BAGNALL, A.J., RAYWARD-SMITH V.J., e WHITTLEY I.M.:"The next release problem."Inform. and Soft. Techn.,vol. 43, 883-890p (2001)

[7] GREER, D., RUHE, G.: Software release planning: an evolutionary and iterative approach, Inform. and Software Technology, 243-253p (2004)

[8] DEB, K., AGRAWAL, S., PRATAB, A., MEYARIVAN, T.: A fast and elitist multiobjective geneticalgorithm:NSGA-II.v.6,n.2,182-197p (2002)

[9] OZERNOY, V. M., Choosing the Best multiple criteria decision making method. INFOR, v. 30, n. 2, p. 159-171 (1992)

[10] TAMANINI, I., Pinheiro, P. R.: Reducing Incomparability in Multiciteria Decision Analysis: An Extension of The ZAPROS Methods, Pesquisa Operacional (Print), v. 31, n. 2, p. 251-270, DOI: 10.1590/S0101- 74382011000200004 (2011)

[11] ZITZLER, E., LAUMANNS, M., THIELE, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. (2001)

[12] BOUYSSOU, D., Marchant, T., Pirlot, M., Perny, P., Tsoukiás, A., Vincke, P.: Evaluation and decision models: a critical perspective. Boton: Kluwer Academic (2000)

[13] FILHO, M. S., PINHEIRO, P. R., ALBUQUERQUE, A. B.: Applying Verbal Decision Analysis to Task Allocation in Distributed Development of Software. SEKE. DOI refer-ence number: 10.18293/SEKE2016-181 (2016)

[14] LARICHEV, O.: Ranking Multicriteria Alternatives: The Method ZAPROS III. European Journal of Operational Research, v. 131, n. 3, p. 550-558 (2001)

[15] TAMANINI, I., PINHEIRO, P. R., MACHADO, T. C. S., ALBUQUERQUE, A. B.: Hy-brid Approaches of Verbal Decision Analysis in the Selection of Project Management Ap-proaches. Procedia Computer Science, v. 55, p. 1183-1192, DOI http://dx.doi.org/10.1016/j.procs.2015.07.093 (2015)

[16] BRASIL, M. M. A., SILVA, T. G. N., FREITAS, F. G., SOUZA, J. T., CORTÉS, M. I.A Multiobjective Optimization Approach to the Software Release Planning with Undefined Number of Releases and Interdependent Requirements. Lecture Notes in Business Inform. Proces, v. 102, p. 300-314, 2012. DOI 10.1007/978-3-642-29958-2_20 (2012)

[17] TAMANINI, I., PINHEIRO, P. R., MACHADO, T. C. S.: Project management aided by verbal decision analysis approaches: a case study for the selection of the best SCRUM practices. Intern. Trans. in Oper. Res., v. 22, p. 287-312, DOI 10.1111/itor.12078 (2015)

[18] BARBOSA, P. A. M., PINHEIRO, P. R., SILVEIRA, F. R. V., FILHO, M. S.: Applying Verbal Analysis of Decision to prioritize software requirement considering the stability of the requirement. 6th Computer Science On-line Conference 2017(CSOC) Advances in Intelligent Systems and Computing, Vol. 575. ISBN: 978-3-319-57141-6. DOI 10.1007/978-3-319-57141-6_45 (2017).

[19] TAMANINI, I., PINHEIRO, P. R.: Challenging the Incomparability Problem: An Approach Methodology Based on ZAPROS. Communications in Computer and Information Science (Print), v. 14, p. 338-347, DOI 10.1007/978-3-540-87477-5_37 (2008)

[20] TAMANINI, I., Machado, T. C. S., Mendes, M. S., Carvalho, A. L., Furtado, M. E. S., Pinheiro, P. R.: A model for mobile television applications based on verbal decision analysis, Advances in Computer Innovations in Information Sciences and Engineering1,1,399–404 (2008)

[21] TAMANINI, I., MACHADO, T. C. S., PINHEIRO, P. R.: Verbal Decision Analysis Applied on the Choice of Educational Tools Prototypes: A Study Case Aiming at Making Computer Engineering Education Broadly Accessible. International Journal of Engineering Education, v. 30, p. 585-595 (2014)

Multi-Objective Crowd Worker Selection in Crowdsourced Testing

Qiang Cui*§, Song Wang†, Junjie Wang*, Yuanzhe Hu*§, Qing Wang*‡§ and Mingshu Li*‡§
*Laboratory for Internet Software Technologies, Institute of Software Chinese Academy of Sciences
†Electrical and Computer Engineering, University of Waterloo, Canada
‡State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
§University of Chinese Academy of Sciences
{cuiqiang, junjie, yuanzhe, wq}@nfs.iscas.ac.cn, song.wang@uwaterloo.ca, mingshu@iscas.ac.cn

***Abstract*—Crowdsourced testing is an emerging trend in software testing, which relies on crowd workers to accomplish test tasks. Typically, a crowdsourced testing task aims to detect as many bugs as possible within a limited budget. For a specific test task, not all crowd workers are qualified to perform it, and different test tasks require crowd workers to have different experiences, domain knowledge, etc. Inappropriate workers may miss true bugs, introduce false bugs, or report duplicated bugs, which could not only decrease the quality of test outcomes, but also increase the cost of hiring workers. Thus, how to select the appropriate crowd workers for specific test tasks is a challenge in crowdsourced testing.**

This paper proposes a Multi-Objective crowd wOrker SElection approach (MOOSE), which includes three objectives: maximizing the coverage of test requirement, minimizing the cost, and maximizing bug-detection experience of the selected crowd workers. Specifically, MOOSE leverages NSGA-II, a widely used multi-objective evolutionary algorithm, to optimize the three objectives when selecting workers. We evaluate MOOSE on 42 test tasks (involve 844 crowd workers and 3,984 test reports) from one of the largest crowdsourced testing platforms in China, and the experimental results show MOOSE could improve the best baseline by 17% on average in bug detection rate.

I. Introduction

Crowdsourced testing is an emerging trend in software testing. In recent years, it has received much attention from the research community [15, 16], and there are many successful crowdsourced testing platforms in industry, e.g., uTest[1], TestBirds[2], and Pay4Bugs[3]. Different from traditional testing, crowdsourced testing entrusts test tasks to crowd workers, who are available on Internet and located in different places. A crowdsourced testing task aims at detecting as many bugs as possible, meanwhile the cost of hiring workers should be controlled within the limited budget.

In current practice, crowd workers search available test tasks themselves and perform any tasks they interested. Thus, test tasks are often performed by a random set of workers. However, for a specific test task, not all crowd workers are qualified to perform it, and different test tasks require crowd workers to have different experiences, domain knowledge, etc. Inappropriate workers may miss true bugs, introduce false bugs, or report duplicated bugs, while hiring them requires nontrivial budget. Selecting inappropriate workers for a test task, not only decreases the quality of test outcome but also increases the cost of hiring workers. Thus, how to select the appropriate crowd workers for specific test tasks is a challenge in crowdsourced testing [9, 19].

The ultimate purpose of crowd workers selection is to select as few workers as possible to detect as many bugs as possible. However, there is no ideal test criterion for this purpose, because the real bug detection information of a test task can only be available until the test task is finished. In this work, we leveraged three alternative criteria for selecting workers, which are critical in crowdsourced testing, i.e., the coverage of test requirement, bug-detection experience of the selected crowd workers, and the cost of the selected crowd workers.

Specifically, *the coverage of test requirement* assures all the test requirements should be tested by the workers with similar expertise. If some test requirements were missed, corresponding bugs might not be detected. This objective is measured using the percentage of terms in test requirement mentioned in workers' historical reports. *The bug-detection experience of the selected crowd workers* assures more experienced workers should be selected to perform the test task, which is important in crowdsourced testing. This is because the experiences of workers vary significantly and experienced workers are more likely to detect bugs [19]. We use the total number of bugs detected by the selected workers to measure this criterion. *The cost of the selected crowd workers* considers the limited budget of a specific test task. Specifically, the cost is the reward for workers, which is set as a constant for each worker performing the test task in most of crowdsourced testing platforms.

In this paper, we propose a Multi-Objective crowd wOrker SElection approach (MOOSE), which aims at selecting the appropriate crowd workers for specific test tasks by considering the above three criteria. MOOSE maximizes the coverage of test requirement, minimizes the cost, and maximizes bug-detection experience of the selected workers. It leverages a widely used multi-objective evolutionary algorithm, namely NSGA-II, to optimize the three objectives when selecting workers.

We evaluate MOOSE on an industrial dataset from Baidu CrowdTest - one of the largest crowdsourced testing platforms

[1]https://www.utest.com/
[2]https://www.testbirds.com/
[3]https://www.pay4bugs.com/

DOI reference number: 10.18293/SEKE2017-102

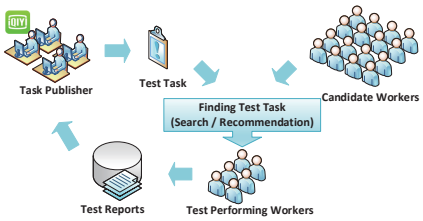


Fig. 1: The typical procedure of crowdsourced testing.

in China. The experimental results show the effectiveness and practical value of MOOSE.

The primary contributions of this paper are as follows:

- We propose a multi-objective crowd worker selection approach, which can maximize the coverage of test requirement, minimize the cost, and maximize the bug-detection experience of the selected workers. **To the best of our knowledge, this is the first approach for the multi-objective crowd worker selection problem.**
- We evaluate our approach on 42 test tasks (involve 844 crowd workers and 3,984 test reports) from one of the largest crowdsourced testing platforms in China. The results show that MOOSE could improve the best baseline on average by 17% in bug detection rate.

II. Background

A. Crowdsourced Testing

The overall procedure of crowdsourced testing is shown in Figure 1. A task publisher provides a test task for crowdsourced testing, including the software under test and test requirements. Then crowd workers could sign in to perform the task, and are required to submit crowdsourced test reports after finishing the task. Finally, the task publisher needs to further inspect the submitted reports and verify whether they are bugs or not.

In current practice, crowd workers search and take proper test tasks all by themselves. However, such process might be ineffective for bug detection, because crowd workers would perform test tasks they are not good at for financial incentive. Consequently, inappropriate workers for a test task may miss true bugs, introduce false bugs, or report duplicated bugs, which could not only decrease the quality of test outcomes, but also increase the cost of hiring workers.

Intuitively, to mitigate the above issue in crowdsourced testing, an appropriate subset of workers should be selected for a specific test task. The selected workers can be recommended to the task publishers, who could then invite them to participate in the task. In addition, the selection results can also assist crowd workers to find proper test tasks so as to save their effort in searching tasks.

In crowdsourced testing, a *test task* is provided by a task publisher, which are described with the natural language test requirements. A *test report* is the test outcome submitted by a crowd worker. It contains the natural language description about operation steps and test outcomes. Specifically, it is labeled by test engineers in the platform to indicate whether the report reveals a "bug", or whether the report is "duplicated" with other reports. A *crowd worker* is a registered worker in the crowdsourced testing platform.

B. Multi-objective Optimization

The multi-objective optimization problem seeks to simultaneously optimize multiple objective functions. It can be defined as to find a vector of variable x, which optimizes a vector of M objective functions $f_i(x)$, where $i = 1, 2, ..., M$.

The multiple objective functions are optimized using Pareto optimality [3], which is a strategy that supposed one player's situation cannot be improved without making the other player's situation worse. Specifically, a decision vector x_1 will dominate a decision vector x_2 if and only if their objective vectors $f_i(x_1)$ and $f_i(x_2)$ satisfy:

$$f_i(x_1) \geq f_i(x_2) \forall i \in 1, 2, ..., M; and$$
$$\exists i \in 1, 2, ..., M | f_i(x_1) > f_i(x_2)$$

All decision vectors that are not dominated by any other decision vectors are called to form the Pareto optimal set, and the corresponding objective vectors are called the the Pareto frontiers.

III. MOOSE

We first present a definition of multi-objective crowd worker selection problem. Given a set of candidate workers W, a vector of M objective functions f_i, where $i = 1, 2, ..., M$, the problem is to find a subset of workers S that is a Pareto efficient set with respect to the objective functions.

To solve the problem, we propose a multi-objective worker selection approach (MOOSE), which maximizes the coverage of test requirement, minimizes the cost, and maximizes the bug-detection experience of the selected workers. Specifically, the cost is the inevitable constraint for worker selection, the other two objectives, i.e., the coverage of test requirement and the bug-detection experience of the selected workers, are critical criteria in crowdsourced testing. We use search-based method to solve the multi-objective optimization. Like other search-based software engineering tasks [4, 7, 10, 13, 20], MOOSE contains representation, fitness function, and computational search algorithm. It also contains a fourth part to illustrate how to handle multiple objectives.

A. Representation

Like other selection problems [4, 6, 14], we encode each worker as a binary variable. If the worker is selected, the value is one; otherwise, the value is zero. The solution is a vector of binary variables, which includes all the candidate workers in crowdsourced testing. The initial population is generated randomly, and the feasible solution would be selected when their values are positive for all objective functions.

B. Fitness

To evaluate the fitness of each solution, we employed a multi-objective function to simultaneously maximize *the coverage of test requirement*, *the bug-detection experience of the selected workers*, and minimize *the cost*.

1) Coverage of test requirement: Test requirement coverage is an important test criterion in software testing. In crowdsourced testing, to cover the test requirements, on the one hand, the selected workers should have similar expertise with the requirements; on the other hand, the workers should be different with each other, so as to cover different parts of the requirements. In this work, we use the conducted crowdsourced tasks and accomplished reports of a worker to represent his/her expertise. Therefore, we use the technical terms in one's historical test reports to represent his expertise. The coverage of test requirement (TR) is measured by the percentage of all terms in the test requirement covered by the expertise of the selected workers. Note that, not all the natural language terms are meaningful for testing, we only use the technical terms obtained based on the method in Section IV-D. Formally, the coverage of test requirement is defined as follows.

$$Coverage = \frac{\textit{\# unique terms of TR mentioned by workers}}{\textit{\# unique terms in TR}} \quad (1)$$

2) Bug-detection experience of the selected workers: In crowdsourced testing, it often has a very distinguished situation that the experiences of crowd workers vary a lot. Some workers have detected many bugs. With the rich experience, they hardly miss true bugs during testing. While some other workers are almost new and do not have much experience, which makes them easily miss true bugs, introduce false bugs, and report duplicate bugs. Thus, it is important to select the experienced workers to perform a specific test task. The bug-detection experience is measured as the total number of bugs the worker detected before. Since duplicated bugs detected by different workers are useless to the overall test outcomes, we use the number of unique bugs detected by a set of workers to represent their bug-detection experience. As mentioned in Section II, duplicated bugs are labeled by test engineers in the platform, thus the unique bugs can be easily picked out.

3) Cost: The cost is an unavoidable objective in crowd worker selection, because the constraint of the cost must be considered when selecting workers in crowdsourced tasks. The straightforward cost in crowdsourced testing is the reward for workers. We suppose all the workers who participate in a test task are equally paid, which is a common practice in real-world crowdsourced testing platforms. The cost is measured as the total reward of the selected workers.

C. Computational Search

We employed a widely used evolutionary algorithm, namely NSGA-II, to simultaneously optimize the three objectives. NSGA-II is a genetic algorithm based optimization technique developed by Deb et al. [3], which is the state-of-the-art optimization technique and has already been widely used in other multi-objective optimization tasks [10].

D. Handling Multiple Objectives

In MOOSE, the three objectives are handled on orthogonal scales. Then we employed Pareto optimality: a solution x_1 is said to dominate another solution x_2, if x_1 is no worse than x_2 in all objectives and x_1 is strictly better than x_2 in at least one objective. According to Pareto optimality, we can search for the set of solutions which are non-dominating.

TABLE I: Running Example.

	Technical terms of TRs of example tasks						Experience (unique bug)				Cost
	t1	t2	t3	t4	t5	t6	b1	b2	b3	b4	value
w_1	√	√							√	√	1
w_2				√	√	√	√	√			1
w_3	√				√	√	√	√		√	1
w_4		√	√	√			√				1

E. Running example

To make MOOSE more clear, we describe it using a running example. As shown in Table I, there are four candidate workers (i.e., w_1 to w_4).

- The workers' historical test reports and test requirements are pre-processed to extract the technical terms. Then for each test task, whether a technical term of test requirements is mentioned by a worker is encoded into a binary vector. For simplicity, we only use six terms as shown in Table I.
- The bug-detection experience of a worker is extracted and encoded into a binary vector. For simplicity, we only use four historical bugs as shown in Table I.
- The cost of selecting each worker is set to "1" and also encoded into a vector.
- The above three binary vectors of a worker are merged into one vector. Then the vectors of workers for a specific test task are passed into the NSGA-II. Through the several iterative generation, selection, and evaluation of NSGA-II, the non-dominating solutions are finally obtained.

In our running example, the solution $\{w1, w2\}$ is dominant $\{w1, w3\}$, because it has a better coverage, a better bug-detection experience, and equal cost. Similarly, $\{w_3\}$ dominates all the solutions containing one worker, $\{w_1, w_2\}$, $\{w_1, w_2, w_4\}$ are the best solutions for selecting two or three workers respectively.

IV. Empirical Study Design

A. Research Questions

- **RQ1**: (Effectiveness of MOOSE) How effective is MOOSE for crowd worker selection?
- **RQ2**: (Necessity of the objectives) Look inside of MOOSE, is each of the three objectives necessary in MOOSE?
- **RQ3**: (Quality of results) Do the results of MOOSE achieve high quality?

B. Evaluation Metric

In this work, bug detection data are used to evaluate the performance of our approach. Given a test task, we measure the performance of a worker selection approach according to whether it can select the "right" workers, who have performed this test task and detected true bugs.

Bug Detection Rate (BDR) is the percentage of bugs detected by the selected crowd workers in a test task out of all bugs historically detected in the same test task. Formally,

TABLE II: Statistics of the dataset used in this work.

Statistic	Number
# of test tasks	42
# of categories	13
# of candidate workers	844
# of test reports	3,984
average # of test reports per task	40
average # of bugs per task	25

given a set of selected workers, i.e., W, and a test task, i.e., T, the bug detection rate is defined as follows:

$$BDR = \frac{\#bugs\ detected\ by\ workers\ in\ W}{\#all\ bugs\ of\ T} \quad (2)$$

Since a smaller subset is usually preferred in crowd worker selection due to the limited budget, we investigate the BDR when selecting from 1% to 50% of the total number of candidate workers for a test task.

C. Baselines

To evaluate the performance of MOOSE, we introduce three baselines. **Random** simulates the current practice in most crowdsourced testing platforms, where workers search test tasks to perform. It randomly selects workers from the candidate set of workers. We run Random for 10 times and record the best performance as its performance.

To further evaluate MOOSE, we also introduce two common ranking baselines, i.e., **Bug Ranking** and **IR Ranking**. **Bug Ranking** ranks all the candidate workers according to the number of historical bugs the worker detected before, and recommends the top workers. **IR Ranking** ranks all the candidate workers according to the textual similarity between the workers' historical test reports and test requirement (Similarity is calculated using Euclidean distance between technical term vectors [18]), and recommends the top workers.

D. Baidu CrowdTest Dataset

We collected crowdsourced testing data from an industrial crowdsourced testing platform, namely Baidu CrowdTest. We collected the test tasks that are closed between Nov. 1st 2015 and Nov. 30th 2015. In total, there are 42 tasks covering 13 categories, such as utilities, lifestyle, finance, etc. For each test task, we manually collected all the detected true bugs of it. The detail statistics of data set are shown in Table II.

To process the dataset, we employed Natural Language Processing to extract a technical term vocabulary for measuring the coverage of test requirements in Section III-B1. Firstly, ICTCLAS[4] is used for word segmentation, then stopwords are removed, and part-of-speech tags are conducted. Finally terms except verbs and nouns are removed. Filtering meaningless terms like existing work [15], we obtain a vocabulary of technical terms.

E. Experimental Setup

To evaluate MOOSE, following existing work [12], we use cross-validation in our experiments. We first randomly selected 70% test tasks as a training dataset and the remaining ones as the test dataset. To mitigate the randomness, the experiment is repeated 20 times, and we use the average BDR to evaluate MOOSE's performance.

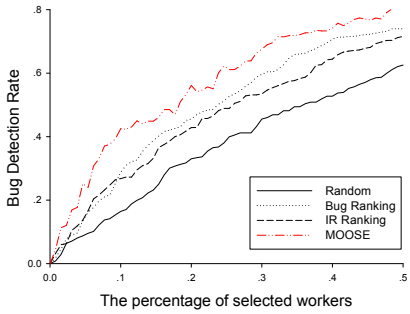


Fig. 2: Comparison of MOOSE with different baseline approaches (RQ1).

For NSGA-II used in MOOSE , we used a random initial population of size 200, and we iterated the algorithm for 200,000 max evaluation, with single point crossover and bit-flip mutation. We set the max evaluation to 200,000, since extended number of evaluations does not show any noticeable improvement in performance. To take into account the inherent randomness of the algorithm, for each test task, we executed 20 independent runs of the algorithm. Note that the running time for NSGA-II is 3,000 ms on average for each test task, which is negligible. We implemented MOOSE based on jMetal[5], a widely used Java framework aiming at solving multi-objective optimization problems.

V. Results

RQ1: How effective is MOOSE for crowd worker selection?

We first compare **MOOSE** with **Random**, which represents the current practice of worker selection in crowdsourced testing platforms. It can be easily observed from Figure 2 that **MOOSE** outperforms **Random**. Specifically, compared with **Random**, the BDR improvement ((MOOSE-Random)/Random) achieved by **MOOSE** is 158% (0.16 vs. 0.42) when selecting only 10% workers, and the BDR improvement is 48% (0.45 vs. 0.67) when selecting 30% workers. In addition, from selecting 1% to 50% workers, the average improvement of BDR is 58%.

Furthermore, **MOOSE** also outperforms both **Bug Ranking** and **IR ranking**, which are two commonly-used ranking methods. The average improvement of BDR for **Bug Ranking** is 17% and for **IR Ranking** is 25%. It can be found that **Bug Ranking** and **IR Ranking** largely outperform **Random**, and in some test tasks, they are close to **MOOSE**. They leverage the measurements that are also about the workers' bug-detection experience and expertise, which implies these two criteria are really useful for selecting workers.

In addition, Table III shows the average BDR when selecting between 1% and 50% workers for each of the 42 test tasks. In most of test tasks (37/42, 88%), **MOOSE** achieves the best performance compared with the three baseline methods. In the

[4]http://ictclas.org

[5]http://jmetal.github.io/jMetal/

TABLE III: The detailed results of MOOSE and other compared methods.

	Bug Detection Rate of all compared methods (RQ1,RQ2)							Bug Detection Rate of all compared methods (RQ1,RQ2)					
	Random	Bug Rank	IR Rank	Bug+Cost	Cov+Cost	MOOSE		Random	Bug Rank	IR Rank	Bug+Cost	Cov+Cost	MOOSE
T1	0.235	0.334	0.274	0.494	0.417	**0.529**	**T22**	0.337	0.498	0.606	0.548	**0.636**	0.631
T2	0.480	0.467	0.413	0.411	0.506	**0.520**	**T23**	0.473	0.492	0.534	0.542	0.492	**0.503**
T3	0.374	0.429	0.518	0.474	0.537	**0.554**	**T24**	0.400	0.617	0.676	0.525	**0.700**	0.659
T4	0.536	**0.677**	0.501	0.667	0.670	0.671	**T25**	0.315	0.150	0.396	0.198	0.492	**0.519**
T5	0.516	0.803	0.810	0.756	0.809	**0.823**	**T26**	0.302	0.584	0.579	0.492	0.635	**0.646**
T6	0.000	0.684	0.292	0.861	0.846	**0.881**	**T27**	0.307	0.405	0.307	0.282	0.452	**0.472**
T7	0.469	0.475	0.381	0.609	0.593	**0.611**	**T28**	0.717	0.620	0.487	0.605	0.703	**0.723**
T8	0.193	0.401	0.204	**0.404**	0.360	0.391	**T29**	0.310	0.117	0.338	0.328	0.220	**0.338**
T9	0.598	0.723	0.560	0.681	0.756	**0.763**	**T30**	0.113	0.009	0.360	0.126	**0.175**	0.107
T10	0.246	0.713	0.600	0.612	0.732	**0.744**	**T31**	0.876	0.909	0.707	0.928	0.723	**0.933**
T11	0.157	0.492	0.500	0.740	0.719	**0.742**	**T32**	0.376	0.390	0.221	0.471	0.344	**0.425**
T12	0.169	0.633	0.083	0.480	0.547	**0.636**	**T33**	0.138	0.000	0.000	**0.169**	0.000	0.153
T13	0.400	0.415	0.451	0.512	**0.569**	0.507	**T34**	0.217	0.378	0.323	0.362	0.347	**0.397**
T14	0.405	0.674	0.623	**0.756**	0.615	0.728	**T35**	0.326	0.325	0.314	0.304	**0.369**	0.345
T15	0.249	0.633	0.486	0.616	0.520	**0.651**	**T36**	0.309	0.493	0.440	0.540	0.575	**0.593**
T16	0.371	0.348	0.225	0.348	**0.389**	0.371	**T37**	0.404	0.419	0.438	0.461	**0.506**	0.436
T17	0.000	0.598	0.576	0.630	0.640	**0.659**	**T38**	0.528	0.579	0.600	0.574	0.771	**0.783**
T18	0.030	0.104	0.070	0.144	0.092	**0.150**	**T39**	0.750	0.764	0.723	0.810	0.707	**0.826**
T19	0.365	0.614	**0.786**	0.587	0.548	0.558	**T40**	0.584	0.510	0.560	**0.676**	0.630	0.609
T20	0.380	0.407	0.368	**0.462**	0.440	0.440	**T41**	0.292	0.617	0.558	0.780	0.817	**0.822**
T21	0.357	0.605	0.619	0.365	0.523	**0.620**	**T42**	0.646	0.476	0.680	0.638	0.668	**0.685**

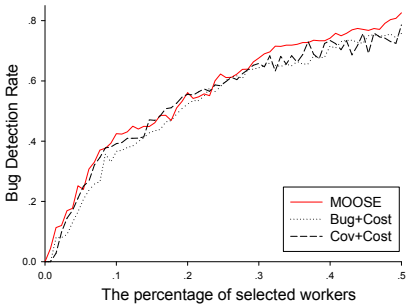


Fig. 3: Comparison between MOOSE and two bi-objective approaches: Bug+Cost and Cov+Cost (RQ2).

rest test tasks, the performance of **MOOSE** is also relatively high.

For all experimental results reported above, we conduct Mann-Whitney U test, and the p-value is much smaller than 0.05. This further implies that **MOOSE** can significantly improve the three baselines.

In summary, **MOOSE** can significantly outperform the current practice in worker selection, as well as two commonly-used ranking methods. Therefore, **MOOSE** is effective for crowd worker selection.

RQ2: Is each of the three objectives necessary in MOOSE?

In crowd worker selection problem, the objective of cost is indispensable, which cannot be removed. Hence, we compare the performance of **MOOSE** when removing either of the other two objectives. We therefore design two bi-objective approaches: **Bug+Cost** (Bug experience of the selected workers+Cost) and **Cov+Cost** (Coverage of the test requirements+Cost).

We can easily see from Figure 3 that our **MOOSE** outperforms the two bi-objective approaches in terms of BDR. The average improvement of BDR for **Bug+Cost** is 9.7%, for **Cov+Cost** is 5.7%.

Table III also presents the average BDRs when selecting between 1% and 50% workers for each of the 42 test tasks. In most of test tasks (29/42, 69%), **MOOSE** is better than both of the bi-objective approaches. In the rest test tasks, the performance of **MOOSE** is also relatively high. For all these experimental results reported above, we conduct Mann-Whitney U test, and the p-value is much smaller than 0.05.

In summary, all the three objectives, which are the coverage of test requirement, bug-detection experience of the selected workers, and cost, are necessary in **MOOSE**.

RQ3: Do the results of MOOSE achieve high quality?

Since MOOSE is a search-based approach, which produces Pareto fronts. To evaluate the quality of Pareto fronts, existing studies in Search-based Software Engineering have applied quality indicators, such as Contribution (I_C), Hypervolume (I_{HV}), and Generational Distance (I_{GD}) [17]. To illustrate the quality of an algorithm, these quality indicators compare the results of the algorithm with the reference Pareto front, which consists of best solutions. To assess the quality of results of MOOSE, we employed these three quality indicators. The set of non-dominated solutions found by **MOOSE**, **Bug+Cost**, and **Cov+Cost** are used as Reference Set (RS) [17].

I_C is the proportion of solutions given by an algorithm that lie on the reference front. The higher this proportion, the more contribution the algorithm to the best solutions and the better the corresponding algorithm. I_{HV} calculates the volume covered by members of a non-dominated set of solutions from an algorithm. The larger this volume, the better the corresponding algorithm. I_{GD} computes the average distance between set of solutions from the algorithm and the reference set. The smaller this distance, the better the corresponding algorithm. Due to the limited space, for details about the three quality indicators, please refer to [17].

For **MOOSE**, we measure these three quality indicators for each test task, then the average values are obtained. The average I_C of **MOOSE** is 0.89, the average I_{HV} is 0.90, and the average I_{GD} is 0.00. The results suggest that the results of **MOOSE** achieve high quality.

VI. Threats to Validity

The threats to external validity concern the generality of this study. First, our experimental data consists of 42 test tasks collected from one of the largest crowdsourced testing platforms in China. The results of our study may not generalize beyond this environment where our experiments were conducted. However, we used different sizes and a variety of data to control this threat. Second, all crowdsourced reports investigated in this study are written in Chinese, and it is not guaranteed that similar results can be observed on crowdsourced projects in other languages. This threat, however, is alleviated as we did

not conduct semantic comprehension, but instead we simply tokenized sentences and used words as tokens for modeling.

The main threat to construct validity in this study involves three objectives in multi-objective formulation. These three objectives are designed from different test criteria: such as the coverage of test requirements, bug detection experience of the selected workers, and cost, but other objectives may also contribute to bug detection in crowdsourced testing. Further exploration of other objectives would address this threat.

VII. Related Work

Crowdsourced testing has been applied to generate test cases [2], solve the oracle problem [11], help usability testing [8], etc. All these studies use crowdsourced testing to solve the problems in traditional software testing activities. There are studies focusing on solving the new encountered problem in crowdsourced testing, e.g., crowdsourced reports prioritization [5] and crowdsourced reports classification [15, 16]. Our approach is also to solve the new encountered and important problem in crowdsourced testing.

The Search Based Software Engineering (SBSE) is an increasingly trend in software engineering. In SBSE, many software engineering problems are reformulated as search problems, such as test case selection [4, 10, 20], and mutation testing [7, 13].

There are several related researches focusing on selecting workers for various software engineering tasks, such as bug triage [6], mentor recommendation [1], expert recommendation [14], etc. All the aforementioned studies either select one worker, or assume the selected set of workers are independent with each other. However, our work selects a set of workers who are dependent on each other, because their performance can together influence the final test outcomes.

VIII. Conclusion

In this paper, we propose MOOSE for crowd worker selection, which maximizes the coverage of test requirement, minimizes the cost and maximizes the bug-detection experience of the selected workers. Experimental results show that MOOSE is effective in bug detection.

In the future, we plan to explore more test criteria that may be helpful for worker selection in crowdsourced testing. Collaborating with Baidu CrowdTest, we are on the way to deploying MOOSE online in order to better evaluate its performance in practice.

IX. Acknowledgments

This work is supported by the National Natural Science Foundation of China under grant No.61432001, No.61602450. We would like to thank the testers in Baidu for their great efforts in supporting this work.

References

[1] G. Canfora, M. Di Penta, R. Oliveto, and S. Panichella. Who is going to mentor newcomers in open source projects? In *FSE'12*, pages 44:1–44:11.

[2] N. Chen and S. Kim. Puzzle-based automatic testing: Bringing humans into the loop by solving puzzles. In *ASE '12*, pages 140–149.

[3] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *ICPPSFN'00*, pages 849–858.

[4] M. G. Epitropakis, S. Yoo, M. Harman, and E. K. Burke. Empirical evaluation of pareto efficient multi-objective regression test case prioritisation. In *ISSTA'15*, pages 234–245.

[5] Y. Feng, J. A. Jones, Z. Chen, and C. Fang. Multi-objective test report prioritization using image understanding. In *ASE'16*, pages 202–213.

[6] G. Jeong, S. Kim, and T. Zimmermann. Improving bug triage with bug tossing graphs. In *FSE'09*, pages 111–120, 2009.

[7] W. B. Langdon, M. Harman, and Y. Jia. Efficient multi-objective higher order mutation testing with genetic programming. *JSS'10*, 83(12):2416 – 2430.

[8] D. Liu, R. G. Bias, M. Lease, and R. Kuipers. Crowdsourcing for usability testing. *ASIS&T'12*, 49(1):1–10.

[9] K. Mao, L. Capra, M. Harman, and Y. Jia. A survey of the use of crowdsourcing in software engineering. *JSS'16*, 126:57–84.

[10] D. Mondal, H. Hemmati, and S. Durocher. Exploring test suite diversification and code coverage in multi-objective test case selection. In *ICST'15*, pages 1–10.

[11] F. Pastore, L. Mariani, and G. Fraser. CrowdOracles: Can the crowd solve the oracle problem? In *ICST'2013*, pages 342–351, March 2013.

[12] F. Sarro, A. Petrozziello, and M. Harman. Multi-objective software effort estimation. In *ICSE '16*, pages 619–630.

[13] R. A. Silva, S. d. R. S. de Souza, and P. S. L. de Souza. A systematic review on search based mutation testing. *IST'17*, 81:19 – 35.

[14] A. Tamrawi, T. T. Nguyen, J. M. Al-Kofahi, and T. N. Nguyen. Fuzzy set and cache-based approach for bug triaging. In *FSE'11*, pages 365–375.

[15] J. Wang, Q. Cui, Q. Wang, and S. Wang. Towards effectively test report classification to assist crowdsourced testing. In *ESEM'16*, pages 6:1–6:10.

[16] J. Wang, S. Wang, Q. Cui, and Q. Wang. Local-based active classification of test report to assist crowdsourced testing. In *ASE'16*, pages 190–201.

[17] S. Wang, S. Ali, T. Yue, Y. Li, and M. Liaaen. A practical guide to select quality indicators for assessing pareto-based search algorithms in search-based software engineering. In *ICSE '16*, pages 631–642.

[18] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[19] Y. Yang, M. R. Karim, R. Saremi, and G. Ruhe. Who should take this task?: Dynamic decision support for crowd workers. In *ESEM'16*, page 8.

[20] S. Yoo and M. Harman. Pareto efficient multi-objective test case selection. In *ISSTA'07*, pages 140–150.

Reuse of Fixture Setup between Test Classes

Lucas Pereira da Silva
Informatics and Statistics Department
Federal University of Santa Catarina
Florianópolis, Brazil
pslucasps@gmail.com

Patrícia Vilain
Informatics and Statistics Department
Federal University of Santa Catarina
Florianópolis, Brazil
patricia.vilain@ufsc.br

***Abstract*—In this paper, we describe commonly used fixture setup strategies as well as their disadvantages and advantages. We propose a dependency model and a test fixture sharing model that allow the definition of a new fixture setup strategy. This strategy promotes code reuse by sharing fixture setups between test classes. The models are evaluated through a case study where the new fixture setup strategy presented a reduction of 47,62% in the fixture setup code.**

Keywords-software testing, unit testing, fixture setup; test fixture; test dependencies; test code reuse

I. INTRODUCTION

Software testing is an important activity of the software development process. The notion of its importance has evolved in recent years. Software testing is no longer an activity that would only start after the coding phase. It is now carried out during the entire development process [4]. In that sense, software testing incorporates other purposes, such as to prevent bug inclusion [1], and to be a way for specification [3] and documentation design [9]. According to [12], tests should be easy to run (automated, self-checking and repeatable), easy to write/understand (simple, expressive and with separation of concerns), and easy to maintain (robust). The satisfaction of these requirements is important to ensure a cost-effective testing activity.

When the software testing starts at early stages of development, some extra challenges appear. According to [11], not only the production code, but also the test code must be maintained. So, it is also necessary to maintain the tests throughout the entire development process [5] [14]. Test maintenance may end up having a high impact on overall testing costs, even higher than the cost of their initial implementation [2]. Therefore, test requirements mentioned above become even more important.

According to [8], the success of automated testing is strongly influenced by the maintainability of the test code. To promote maintainability, the tests should be clearly structured, well named and small in size [8]. Furthermore, code duplication across the tests must be avoided [8]. Tests that are hard to write and understand may suggest a poor system design.

According to [7], each test case is usually separated in a test method. A test case simulates a use scenario of the SUT. In this sense, a test case has to put the SUT in a state that represents the use scenario, that is, a state of interest to the test [11]. This is done through the execution of the test fixture setup code. According to [12], a test fixture represents anything necessary to exercise the SUT. The code where test fixtures are created is called fixture setup.

Reference [12] presents several strategies for fixture setups. Among these strategies, we highlight the inline setup, implicit setup and delegate setup. Each strategy has a different impact in the following properties of the test code: simplicity (tests should be small and verify one thing at a time), expressiveness (tests should communicate intention, i.e., should be easy to understand what behavior a test wants to verify), separation of concerns (each test should focus only on a single concern of the system) and robustness (overlap between tests should be avoided in such a way that small changes on the production code do not affect a large amount of tests). It is important to note that the robustness is directly influenced by code duplication. Thus, avoiding the duplication of test code can help to reduce the development effort and, consequently, to create tests with better maintainability.

The objective of this work is to propose a new fixture setup strategy that may be used as an option to promote better robustness of the testing code without affecting its other properties. We also propose a model to represent this new fixture setup strategy. Thus, test frameworks may incorporate this fixture setup strategy by implementing the proposed model. It is important to point out that our intention is not to propose a strategy that is superior to the others, but to present a new strategy that helps developers to create better tests with less effort.

The sections of this paper are organized as follows: in Section 2 we present the most known fixture setup strategies in the literature; in Section 3 we propose a model that allows the reuse of fixture setups between test classes; in Section 4 we present a case study; in Section 5 we evaluate, through the use of framework Story implemented, the proposed model; and in Section 6 we present the conclusions of our work.

II. FIXTURE SETUP STRATEGIES

Meszaros [12] introduces the most used fixture setup strategies, mainly in the frameworks of xUnit family. The strategies are categorized as fresh fixture setup, where the fixture setup is run before at each test, or as shared fixture construction, where the fixture setup is run only once before a given group of tests. Both categories have strategies that allow the reuse of fixture setups (i.e., reuse of code). However, only

10.18293/SEKE2017-181

in the shared fixture construction is possible to reuse test fixtures (i.e., reuse of execution).

A. Fresh Fixture Setup

The fresh fixture setup category is composed by three strategies: inline setup, implicit setup and delegate setup. According to [12], an adequate combination of these three strategies may increase the code reuse and, even so, preserve the desired properties of simplicity, expressiveness and separation of concerns. Next, these three strategies are presented, as well the main advantages and disadvantages of each one.

1) Inline Setup: in this strategy the fixture setup is placed inside the test method. In general, this strategy is chosen when a test needs a very specific test fixture or during the beginning of the development of the test code when duplication of code does not exist yet. An advantage of this strategy is a better understanding of the relationship between the test fixtures and the behavior that the test is verifying, since the fixture setup is near to the test verifications. However, its major disadvantage is the duplication of code among test methods.

2) Implicit Setup: In this strategy the fixture setup is placed in a special method (usually called setup method) of the test class. The test framework is responsible for running the setup method before the execution of each test method existing in the test class. The reuse of the fixture setup is the major advantage of this strategy. However, higher is the amount of test methods in the same test class, harder is to preserve the properties of simplicity, expressiveness and separation of concerns. When many test methods are created, it is more difficult to understand the relationship between the test fixtures and the intention of each test, once some test fixtures may not be needed to all tests.

3) Delegate Setup: in delegate setup, the code is placed in helper methods. In general, these methods are put in auxiliary classes, apart the test methods. Promoting fixture setup reuse between distinct test classes is the major advantage of this strategy. However, this strategy may demand high labor. Helper methods and classes must be created and maintained by developers. Thus, well-named helper methods are essential in order to preserve the expressiveness. Furthermore, in contrast with inline setup and implicit setup strategies, the framework does not call automatically the delegate setup.

B. Shared Fixture Setup

The strategies of the shared fixture category depend on a place to save test fixtures that are created. This place may be a file system, a database or even a static field. The fixture setup runs once for a given group of tests and then the created test fixtures are saved. Thus, the tests of that given group may access the test fixtures. The definition of when the fixture setup will be run varies according to each strategy.

III. PROPOSAL

Test code duplication is related to the robustness property, which in its turn is related to test code maintainability. Thus, reducing the test code duplication may improve the test code maintainability without impacting negatively in simplicity, expressiveness and separation of concerns. Strategies as implicit setup and delegate setup help to reduce the test code duplication. However, sometimes, to reduce the duplication of test code without affecting other properties may be a hard task. Higher is the amount of tests, harder is to maximize the test code reuse by clustering test classes according to the test fixtures (implicit setup) and harder is to maximize the test code reuse by creating many helper methods as test fixtures needed (delegate setup).

In this work we propose the definition of a model to represent the dependence between test classes in order to create a new fixture setup strategy. The proposed model allows the reuse of fixture setups between different test classes. The main objective is to increase the tests robustness by increasing the reuse of fixture setups without reducing the general tests maintainability.

The central assumption of our proposal comes from the simple observation that a given fixture setup execution may drives the SUT exactly to a specific state of another test class. It is crucial to clarify that the explicit definition of dependence between test classes, proposed by this work, does not break the independence principle described in [13].

A. Dependency Model between Test Classes

The dependency model between test classes considers that a test class may depend on one or more test classes. The definition of dependence must be done in the dependent class. The dependency/dependent relationship between two test classes means that the dependency class has the fixture setup required by the dependent class.

Fig. 1 shows a hypothetical implementation of the dependency model, using the Java language. The annotation @FixtureSetup is placed in the declaration of the test class UserTest (dependent) to indicate a dependency with the test class CleanDatabaseTest (dependency). Thus, each test run of the dependent class should be preceded, in the given order, first by the run of the fixture setup of the class CleanDatabaseTest and then by the run of the fixture setup of the test class UserTest. This enables the reuse of fixture setup between different test classes.

```
@FixtureSetup(CleanDatabaseTest.class)
class UserTest { ... }
```

Figure 1. Annotation @FixtureSetup.

B. Independence Principle

Independence principle, described in [13], says that each test should be independent. It should be possible to run each test individually or through a suite of tests in an arbitrary order. This principle is justified by the fact that a given test run should not affect another test run. If a test fails because a previously test had let the SUT in an inconsistent state, then the independence principle is violated.

It is important to clarify that the proposed dependency model does not violate the independence principle. The dependency model implies only in dependence between test classes, but not between tests. In the dependency model, before

each test run, the test framework has to run all the needed fixture setup.

C. Multiple Dependencies

Fig. 2 shows an example where a test class depends on two test classes. Fixture setup run will follow the order in which the dependency classes are declared (i.e. the order of the dependency classes in the annotation @FixtureSetup).

Fixture setups defined in different test classes can be combined to construct a new fixture setup. In the dependency model, this combination is achieved without any change in the dependency classes.

```
@FixtureSetup({
 UserTest.class,
 EventTest.class
})
class UserEventTest { ... }
```

Figure 2. Multiple dependencies in annotation @FixtureSetup.

D. Multiple Dependencies

The dependence relationship between test classes is transitive. Thus, it is possible to create a chained sequence of test classes. Each test run of a given test class will be preceded by the recursive execution of the fixture setups.

Test class sequences may be especially useful for implementation of evolutionary acceptance test specifications. Acceptance tests, also known as story tests [10], are user tests that verify if a system satisfies the acceptance criteria defined by a client [3]. In an evolutionary specification, the automated acceptance tests model use scenarios of the system where each test has as start point in a previously scenario [6].

E. Test Fixture Sharing Model between Test Classes

In the dependency model, a test class may reuse the fixture setup of another test class. The fixture setup of a given test class is the code that prepares the SUT in order to run the tests included in the test class. Test fixtures are the elements necessary to the tests and are created by the fixture setup run, such as: an object, a record in a database, a file, etc.

The nature of both test fixture and fixture setup strategy determines the way in which the test fixture is available to the test. In the inline setup, implicit setup and delegate setup strategies the ways the test fixture are available are, respectively, a local variable, a test class and a return of the helper method call.

The contributions of the work proposed in [11] were used here. That work presents a tool, called Picon, to promote code reuse in an isolated file. Each fixture setup is associated with a qualifier. The fixture setup run consists in injecting an object in a field (named with the same qualifier) of the test class. The injected object contains the test fixtures described in the fixture setup.The qualifier should be simple and, at the same time, express the fixture setup configuration [11]. Thus, the authors claim that the developer may quickly identify a given fixture setup just reading the qualifier.

Besides the dependency model, we also define a test fixture sharing model between test classes. The purpose of this model is to make the test fixtures of a given test class available to tests of a different test class. A sharing mechanism allows sharing class fields and test fixture qualifiers. So, tests of a dependent class may access a test fixture associated to a field in a dependency class through the declaration of a field named with the desired test fixture qualifier created in the dependent class. Thus, the test framework must inject the field of the dependency class in the field of the dependent class.

F. Graph Model

In the dependency model, each test class can have as many dependency classes as needed. Furthermore, the dependency relationship is transitive. Thus, a directed graph is defined to represent test classes and dependency relationships. The digraph $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$ is defined as follow:

$$\boldsymbol{V} = \{c \mid c \text{ is a test class}\}$$

$$\boldsymbol{E} = \{(c, d) \mid c \text{ directly depends on } d\}$$

We call $\mathcal{G}$ as Static Dependence Graph (SDG). The set $\boldsymbol{V}$ contains all test classes of the system. However, sometimes it is convenient to include only test classes involved in a given test run. Therefore, the subgraph $\mathcal{H} = (\boldsymbol{W}, \boldsymbol{F})$ is defined from the SDG as follow:

$$\boldsymbol{W} = \{c \in \boldsymbol{V} \mid c \text{ is a test class of the running test or } c \text{ is a dependency class of the running test}\}$$

$$\boldsymbol{F} = \{(c, d) \mid c \text{ directly depends on } d\}$$

We will call $\mathcal{H}$ as Dependence Execution Graph (DEG). Both SDG and DEG are digraphs. Thus, the arrow points from the dependent class to the dependency class. In the DEG of the Fig. 3, the vertex $\boldsymbol{\mathcal{TD}}$ is the test class of the running test. The $\boldsymbol{\mathcal{TA}}$, $\boldsymbol{\mathcal{TB}}$ e $\boldsymbol{\mathcal{TC}}$ vertices are the dependency classes of $\boldsymbol{\mathcal{TD}}$. A depth first search starting from $\boldsymbol{\mathcal{TD}}$ gives the Fixture Setup Execution Sequence (FSES) needed for the running test. The FSES gives the order in which the fixture setups should be run to prepare the SUT for the test. The depth first search can visit a vertex more than once. There are two valid FSES for the DEG shown in Fig. 3:

$$\mathbf{S1} = (\boldsymbol{\mathcal{TA}}, \boldsymbol{\mathcal{TB}}, \boldsymbol{\mathcal{TA}}, \boldsymbol{\mathcal{TC}}, \boldsymbol{\mathcal{TD}})$$

$$\mathbf{S2} = (\boldsymbol{\mathcal{TA}}, \boldsymbol{\mathcal{TB}}, \boldsymbol{\mathcal{TC}}, \boldsymbol{\mathcal{TD}})$$

In $\mathbf{S1}$ the approach includes each vertex $\boldsymbol{v}$ whenever $\boldsymbol{v}$ is found. So, the fixture setup run of $\boldsymbol{\mathcal{TA}}$ is included twice. In other side, in $\mathbf{S2}$ the approach includes each vertex $\boldsymbol{v}$ only the first time that $\boldsymbol{v}$ is found.

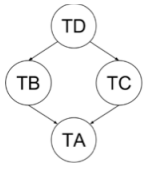


Figure 3. Dependence Execution Graph.

G. Oneness of Fixture Setup

A given fixture setup has the oneness property when it should run only once during a same test run. To repeat a fixture setup run when the oneness property is true can cause an unwanted failure in the test. The dependency model will add

two new constraints: (1) the default value for the oneness property of a fixture setup is false; and (2) if the oneness property of a fixture setup is true, then it should be explicitly declared in the test class. In the framework Story, to define as true the oneness property we have to annotate the test class with the annotation @Singular.

IV. CASE STUDY

This section presents a case study where the proposal of this work was applied. Fig. 4, 5, 6 and 7 present test classes of a system for event scheduling and tracking. Many users can participate of one event and each user should inform his available schedule for the given event. Only the tests for the entities User and Event where considered in this case study. The test classes used in the case study were specifically chosen in order to exemplify the most important aspects of the proposal. Our intention is to show how the approach could be applied in a real development scenario.

The tests for the entities User and Event are included in test class UserTest (Fig. 4) and test class EventTest (Fig. 5), respectively. The test class UserEventTest (Fig. 6) contains the tests for the associative entity UserEvent.

The fixture setup UserTest is defined by the implicit setup method createAndInsertJohn(). Both tests of the test class UserTest depend on having the test fixture john registered in the database. The registration of the test fixture john is done by the UserTest fixture setup.

```
@FixtureSetup(NoDataTest.class)
class UserTest {
 @Fixture UserDao userDao;
 User john;
 Long id;

 @Before void createAndInsertJohn() {
  john = new User();
  john.setName("John");
  john.setCareer("Teacher");
  id = userDao.insert(john);
  assertNotNull(id);
 }

 @Test void get() {
  User user = userDao.getEntity(id);
  assertEquals(id,user.id());
  ... }

 @Test void list() {
  List<User> list = userDao.list();
  ...}
}
```

Figure 4. UserTest test class.

The tests of the test class UserTest have, as starting point, a scenario where the database is empty. This scenario is expected in order to have a greater control over the tests and to avoid fragile tests [12]. There are two common approaches to put the SUT in the expected starting scenario: (1) clean all database before each test run; or (2) remove the persisted entities after each test run. The second approach could be achieved with the use of the @After JUnit annotation. This annotation is analogous to @Before but the difference is that @After is run after the test method. To simplify the study case, we do not take into account the use of the @After annotation in this work. The annotation, however, could be easily applied to the dependency model. To do that, it would be only necessary to run the methods annotated with @After in the inverse order of the methods annotated with the @Before.

```
@FixtureSetup(NoDataTest.class)
class EventTest {
 @Fixture EventDao eventDao;
 Event lecture;
 Long id;

 @Before void configure() {
  lecture = new Event();
  lecture.setName("Lecture");
  id = service.insert(lecture);
  assertNotNull(id);
 }

 @Test void get() {
  Evento event = eventDao.entity(id);
  assertEquals(id,event.id());
  assertEquals(lecture.id(),event.id());
  assertEquals("Lecture",event.name());
}

 @Test void list() {
  List<Event> list = eventDao.list();
  ... }
}
```

Figure 5. UserTest test class.

The test class NoDataTest (Fig. 7) contains tests that verify the SUT when the persistence layer is empty. For this, the fixture setup of the test class NoDataTest removes all possible entities persisted in the database. It is noteworthy that the starting point expected by the test class UserTest is exactly the state of the SUT after running the fixture setup of the test class NoDataTest. Initially, two strategies are considered in order to promote code reuse: (1) move the tests of the test class UserEventTest to the test class NoDataTest; or (2) extract the fixture setup of the test class NoDataTest to a helper class and promote the code reuse through a delegate setup. The first approach has some limitations regarding the organization of the test classes. Besides this, the implicit setup method createAndInsertJohn cannot be used because it would cause an unwanted failure in the test method emptyData(). The second approach requires an effort to create auxiliary code artifacts. Furthermore, the approach can contribute with obscure tests, since the fixture setup of the test class NoDataTest should be moved to a different place from that where it was originally defined. This makes harder to understand the relationship between the test fixture and the expected behavior of the test.

Using the dependency model we are proposing, the test class UserTest can reuse the fixture setup NoDataTest without affecting the structure of the involved test classes. In the example, this is achieved using the annotation @FixtureSetup. The annotation @FixtureSetup indicates to the test framework that the run of the fixture setup of the test class NoDataTest should be done before each test run of the test class UserTest. We highlight the fact that the test fixture userDao is created in the fixture setup NoDataTest and is shared with the tests of the test class UserTest through the class fields. The class field UserTest.userDao is annotated with the annotation @Fixture and is named according to the test fixture qualifier userDao. The test fixture userDao is associated to the class field NoDataTest.userDao during the run of the fixture setup NoDataTest. After that, the test framework has to inject the class field NoDataTest.userDao into the class field UserTest.userDao.

The test class EventTest is analogous to the test class UserTest, then the same considerations about the test class UserTest are also applied to the test class EventTest.

The tests of the test class UserEventTest depend on the test fixtures john and lecture. The test fixture john is created in the fixture setup UserTest and the test fixture lecture is created in the fixture setup EventTest. Using the dependency model, the test class UserEventTest can incorporate the needed fixture setups and, thus, increase code reuse.

```
@FixtureSetup({
 UserTest.class,
 EventTest.class
})
class UserEventTest {
 @Fixture User john;
 @Fixture Event lecture;
 @Fixture UserEventDao dao;
 UserEvent johnLecture;
 Long id;

 @Before void configure() {
  johnLecture = new UserEvent();
  johnLecture.setUser(john);
  johnLecture.setEvent(lecture);
  id = dao.insert(johnLecture);
  assertNotNull(id);
 }

 @Test void get() {
  UserEvent entity = dao.entity(id);
  ...   }

 @Test void list() {
  List<UserEvent> list = dao.list();
  ... }
}
```

Figure 6. UserTest test class.

The test class NoDataTest has a particularity regarding the other test classes. Its fixture setup has the oneness property (i.e. the fixture setup run should not repeat for a same test run). The annotation @Singular is used in order to declare the oneness property of the fixture setup NoDataTest. The absence of the annotation causes an unwanted failure at the tests of the test class UserEventTest because the NoDataTest fixture setup run would be run twice: before the fixture setup UserTest and before the fixture setup EventTest. In this case, the second run of fixture setup NoDataTest would remove the test fixture john (persisted in the UserTest fixture setup run). The annotation @Singular prevents the test framework from repeating the NoDataTest fixture setup run, avoiding the test fixture john be removed.

```
@Singular
class NoDataTest {
 EventDao eventDao;
 UserDao userDao;
 EventUserDao dao;

 @Before void configure() {
  eventDao = new EventDao();
  userDao = new UserDao();
  dao = new EventUserDao();
  eventDao.removeAll();
  userDao.removeAll();
  dao.removeAll();
 }

 @Test void emptyData() {
  assertTrue(eventDao.list().isEmpty());
  assertTrue(userDao.list().isEmpty());
  assertTrue(dao.list().isEmpty());
 }  }
```

Figure 7. UserTest test class.

In the case study, we observe that the dependency model and the test fixture sharing model facilitate the development of tests in an iterative and evolutionary approach. It allows reusing fixture setups between test classes without affecting their structures.

V. EVALUATION

In order to evaluate the applicability of the dependency model and the test fixture sharing model we realized an experiment. The aim of the evaluation was to identify any difference between the code reuse of a set of tests using the conventional fixture setup strategies and the code reuse of a set of equivalent tests using the proposed fixture setup strategy. This experiment was realized through the development of the system presented in Section IV.

The experiment was conducted as follow. First of all, tests were developed following an iterative development. The tests should run in the framework JUnit and use the conventional test fixture setup strategies. The conclusion of the system resulted in 77 tests and 11 test classes. A subset, composed by 24 tests and 4 test classes, was selected from the total tests. This subset was named as control group. The tests selected to be included in the control group should include real database operations. Then, the control group was manually rewritten in order to use the dependency model and the test fixture sharing model of the framework Story, that implements our proposal. The set of rewritten tests, named as experimental group, was composed by 24 tests, 14 test classes and 1 helper class. The creation of the experimental group had consider the following constraints: (1) for each test of the control group should exist an equivalent test in the experimental group; (2) the test coverage should not change between the two groups; (3) the test fixture set and assertion set for each test of the control group should be the same for the equivalent test in the experimental group; (4) names of variables, methods and class fields should be preserved whenever possible; (5) Story annotations should be placed in an individual line; and (6) the tests can be freely reorganized since the previous restrictions are respected.

After the experiment, the follow measurements were collected in the control group and experimental group: (1) amount of code lines of test classes and helper classes; (2) sum of the amount of repeated lines, excluding assertions; (3) sum of distinct repetitions, excluding assertions; (4) sum of the amount of repeated lines, including assertions; and (5) sum of distinct repetitions, including assertions. In all measurements we ignored: blank lines, package declarations and import declarations. In the measurements 2, 3, 4 and 5 the following symbols were not counted as repetitions: annotation @Test, annotation @Before, annotation @Fixture, identic method declaration and block delimiter symbol.

Fig. 8 shows the results for the execution of the three distinct test groups: (a) tests of the control group; (b) tests of the experimental group; and (c) tests of both groups. As we can see in Fig. 8 (a) and in Fig. 8 (b), the test execution time between the two groups is consistently alike. This behavior is a reasonable proof that the third restriction was not violated.

The experiment results are shown in Fig. 9. Each time reported is the result of a single execution. Black bars represent control group measurements while gray bars represent the experimental group measurements. The experimental group had an amount of test code lines slightly larger. The extra annotations of Story can justify the increase in the amount of test code lines. However, the experimental group presented a

considerable reduction in the amount of repeated lines (126 for the control group and 66 for the experimental group). Considering the proportionality of test code lines, the control group had 40,91% of repeated lines while the experimental group had 20,37%. Comparing the control group with the experimental group, the last one had, considering absolute values, a reduction of 47,62% of the repeated lines.

Figure 8. Execution of test groups through the Story.

VI. CONCLUSION

This work focused on presenting models to help increasing code reuse without negatively affecting code simplicity, expressiveness and separation of concerns.

The dependency model and test fixture sharing model presented in this work contribute with a new fixture setup strategy. Through this strategy, it is possible to promote test code reuse without losing the freedom of reorganizing test classes.

Furthermore, we presented a study case where we qualitatively evaluate how code simplicity, expressiveness and separation of concerns are well preserved. Another contribution was the definition of the oneness property applied to fixture setups.

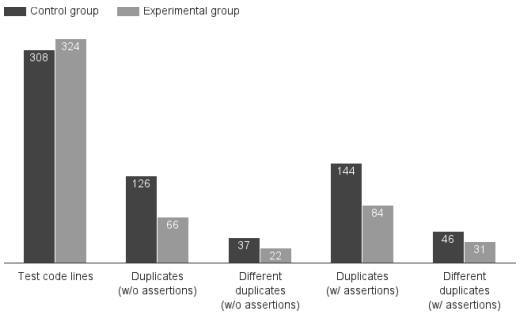


Figure 9. Experiment results.

REFERENCES

[1] Beizer, B. 1990. *Software Testing Techniques (2nd ed.)*. Van Nostrand Reinhold Co., New York, NY, USA.

[2] Berner, S., Weber, R., and Keller, R.K. 2005. *Observations and lessons learned from automated testing*. In Proceedings of the 27th International Conference on Software engineering (ICSE '05). ACM, New York, NY, USA, 571-579.

[3] Bertolino, A. 2007. *Software Testing Research: Achievements, Challenges, Dreams*. In Procedings of the 2007 Future of Software Engineering (FOSE '07). IEEE Computer Society, Washington, DC, USA, 85-103.

[4] Borg, R., and Kropp, M. 2011. *Automated acceptance test refactoring*. In Proceedings of the 4th Workshop on Refactoring Tools (WRT '11). ACM, New York, NY, USA, 15-21.

[5] Canfora, G., Cimitile, A., Garcia, F., Piattini, M., and Visaggio, C.A. 2006. *Evaluating advantages of test driven development: a controlled experiment with professionals*. In Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering (ISESE '06). ACM, New York, NY, USA, 364-371.

[6] Freeman, S., and Pryce, N. 2009. *Growing Object-Oriented Software, Guided by Tests (1st ed.)*. Addison-Wesley Professional.

[7] Zaidman, A., Deursen, A., and Storey, M.A. 2013. *Strategies for avoiding text fixture smells during software evolution*. In Proceedings of the 10th Working Conference on Mining Software Repositories (MSR '13). IEEE Press, Piscataway, NJ, USA, 387-396.

[8] Hanssen, G.K., Haugset, B. 2009. *Automated Acceptance Testing Using Fit*. In Proceedings of the 42nd Hawaii International Conference System Sciences (HICSS '09). IEEE Computer Society, 1-8.

[9] Kamalrudin, M., Sidek, S., Aiza, M.N., and Robinson, M. 2013. *Automated Acceptance Testing Tools Evaluation*. In Agile Software Development. Sci. Int, 4, 1053-1058.

[10] Longo, D.H., Wilges, B., Vilain, P., and Cislaghi, R. 2015. *Fixture Setup through Object Notation for Implicit Test Fixtures*. Journal of Computer Science 11, 6, 794.

[11] Meszaros. G. 2006. *xUnit Test Patterns: Refactoring Test Code*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[12] Meszaros, G., Smith, S., and Andrea, J. 2003. *The test automation manifesto*. In Proceedings of the 3rd XP Universe Conference (XP'03). New Orleans, LA.

[13] Pinto, L.S., Sinha, S., and Orso, A. 2012. *Understanding myths and realities of test-suite evolution*. In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (FSE '12). ACM, New York, NY, USA, Article 33, 11 pages.

[14] Tiwari, R., and Goel, N. 2013. *Reuse: reducing test effort*. SIGSOFT Softw. Eng. Notes 38, 2 (March 2013), 1-11.

An Ontology-based Knowledge Management System for Software Testing

Shanmuganathan Vasanthapriyan[†], Jing Tian[*], Dongdong Zhao[‡],Shengwu Xiong[§] and Jianwen Xiang[*]
Hubei Key Laboratory of Transportation Internet of Things
School of Computer Science and Technology
Wuhan University of Technology
Wuhan, P.R. China
Email: priyan@appsc.sab.ac.lk , {jtian*,zdd,xiongsw,jwxiang*}@whut.edu.cn

***Abstract*—Software testing is an important activity in quality assurance and it generates large amount of knowledge. Software testers need to gather domain knowledge to be able to successfully conduct a software testing activity. Not having a proper knowledge base within its own context by software testing environments cause software testers to query limited knowledge available or consult peer software testers, which would greatly impact on their decision-making process. Ontologies emerge as one of the more appropriate knowledge management tools for supporting knowledge representation, processing, storage and retrieval. Given great importance to knowledge for software testing, and the potential benefits of managing software testing knowledge, using semantic web technologies, ontology based knowledge management system is developed. A Software testing knowledge sharing ontology is designed to describe software testing domain knowledge. SPARQL is used as the query language to retrieve software testing knowledge from the semantic storage. Both Ontology experts and non-experts evaluated the developed ontology. We believe our software testing ontology can support other software organizations to improve the sharing of knowledge and learning practices.**

***Keywords*—software testing ontology, software testing knowledge, ontology based knowledge management system, knowledge sharing.**

[1] [2]

I. Introduction

Software testing is a sub area of software engineering which is also a knowledge intensive and collaborative activity[1][2]. Knowledge can be applied to different testing tasks and purposes. Since software development is an error prone task, to achieve quality software products, Validation and Verification should be carried throughout the development[3]. As software testers, they would be familiar with the several software testing methods and considerably aware of the software development models, need information relevant to their context. For instance, software testers may either need assistance on a test case design information relevant to a similar project which was handled previously for testing purposes or to design a test case. Moreover, this information would also have a greater impact on their decision-making process. Importantly, during the software testing activities a huge amount of knowledge is being continuously produced and consumed. But, the approaches are limited and less employed to capture this knowledge and manage inside the organization to facilitate software testers. Our previous study results revealed some of the issues such as, outdated knowledge in the repositories, un structured internal documents and varied formats, less accessing facilities and lack of targeted delivery methods. Hence, efficient mechanisms for capturing, representing, reusing, and sharing the software testing knowledge involved are sorely needed.

According to Gruber[4], an ontology is an explicit specification of a conceptualization. Ontology provides a structured view of domain knowledge and acts as a repository of concepts in the domain. Recently, ontologies and Semantic Web technologies have received more attention and been gradually used in the knowledge representation[5]. Given great importance to knowledge for software testing, and the potential benefits of managing software testing knowledge, using semantic web technologies, ontology based knowledge management system is developed.

The context has also been decided to confine the study to a particular Sri Lankan software development company. The key reasons are based on the geographical location of the researcher, practicality and ease of access to those software development companies and comparability of research data due to company's same jurisdiction, same economic and regulatory regimes governing their operation. Further, we briefly explain each type of the high-level concepts based on IEEE 829-2008[6], also known as the 829 Standard for Software and System Test Documentation and ISTQB (International Software Testing Qualifications Board)[7]. Even though the standard specifies the procedures of software testing, we have also included what software company is stipulating in it's practice.

The remainder of this paper is organized as follows. Succinct analysis of related research is presented in the second section. Section 3 discusses software testing ontology in detail. Development of knowledge management portal to manage software testing knowledge is discussed in Section 4. Section 5 discusses the evaluation of the ontology developed under two points-of-view: domain experts and non-experts. Finally, Section 6 concludes the paper and presents directions for future work.

[1]DOI Reference Number: 10.18293/SEKE2017-020
[2]*Corresponding Authors

II. Related Work

A number of overviews of work on knowledge management in software testing have previously been published. The research carried out by Wei and Ying[8] discussed the proposal of implementing test knowledge management framework in iDEN phone software system testing and how the knowledge management approach can benefit the testing team in terms of cost and productivity. A reusable test case knowledge management model is proposed by Li and Zhang [9] to support the knowledge reuse based on the ontology representation of reusable test cases so that the test engineers can retrieve and reuse test cases flexibly. Douglas [10] proposed to investigate an open-standard based approach to the sharing of test results in the form of digital objects. Such an approach would not only reduce the needless replication of tests that occurs when there are no public records of previous tests conducted, but it would also allow the accumulation of a good deal of evidence to support certain usability design patterns and guidelines. Reference Ontology on Software Testing (ROoST) which was developed managing relevant knowledge to reuse is difficult and it requires some means to represent and to associate semantics to a large volume of test information[11].Bajnaid[12] proposed an SQA ontology that represents both domain and operational knowledge which provides consistent support to communicate between people and software agents but does not eliciting anything related to software testing process.

III. Designing of Software Testing Domain Ontology

A. Describing the Software Testing Process

Software testing process contains *TestPlanning* for planning tests, *TestCaseDesign* for test case construction, and test execution *TestExecution* for execution of test cases and producing *ActualResult*, and *TestResultAnalysis* for analysis and evaluate the test results [13]. In addition, our software testing process includes *Test Design Techniques*, *Test Levels*, *Artifacts*, *TestEnviorment* (includes hardware, software and Human resources) and *Static Testing Techniques*.

A *TestPlan* is produced during the *TestPlanning* activity. TestCaseDesign activity targets to design a *TestCase*. During the *TestCaseDesign*, if a *Test Design Techniques* is used then the following axiom can be defined for *TestCaseDesign* to design any *TestCase* as follows.

$\forall$ *tc:TestCase, tdt:TestDesignTechnique , tcd:TestCaseDesign hasDesignAccordingTo(tc,tdt) and isGeneratesTestCase(tcd,tc)* $\longrightarrow$ *isAdopts (tcd,tdt)*

Several Artifacts have been used to derive test cases in software testing which describes the functionalities, architecture, and design of software. Such Artifacts are used as *TestCaseDesignInput* during the software *TestCaseDesign* activity and the output is test cases. Test cases can be documented as described in the IEEE 829-2008 documentation. The document that describes the steps to be taken in running a set of tests (and specifies the executable order of the tests) is called a test procedure in IEEE 829. Besides, Test case contains a set of input values (*TestCaseInput*), execution preconditions, expected results (*Expected Result*) and execution post conditions, developed for a particular objective or test condition, such as to exercise a particular program path or to verify compliance with a specific requirement.

That is, a test case targets to test a *Code To be Tested* . *Code To be Tested* can be any programs, modules, and the whole system code. Furthermore, *Test Code* is a portion of code that is to be run for executing a given set of test cases, contain three subtypes such as *Test Script, Driver* and *Stub*.

Test Execution activity executes any Test case. Test Execution requires as input the *Test Code* to be run and the Code To Be Tested. Notably, both *Test Code* and *Code To Be Tested* are needed for Test Execution activity. This can be illustrated in an axiom as follows.

$\forall$ *te:TextExecution, tc:TestCase hasExecutesTestCases(te,tc)* $\longrightarrow$ *(* $\exists$ *tcode:TestCode, CodeTobeTest: CodeToBeTested) uses (te,tcode)* $\bigwedge$ *isContainedOf(tcode,tc)* $\bigwedge$ *uses (te,CodeTobeTest)* $\bigwedge$ *hasCodeTobeTested(tc,tcode)*

The output of the *Test Execution* activity is the *Test Result*. Each of the *Test Result* is particularly related to a TestCase. *Test Result* may be related to an *Actual Result*, for a particular *TestCaseInput* and *ExpectedResult*. *Test Execution* also can be elaborated necessary and sufficient axioms in Protégé 5.1 as follows.

hasExecutesTestCases only (TestCase and (hasTestCaseExpectedResult some ExpectedResult))

A test execution can run and achieve a result (*Actual Result*), but it can also fail, and generating a *Test Issue* . Thus, a Test Result contains either an *Actual Result*, or *Test Issue* or both. This can be expressed using the following axiom.

$\forall$ *TestR:TestResult* $\longrightarrow$ *(*$\exists$ *ActResult:ActualResult, Issue: TestIssue) (ActualResult(ActResult)* $\bigvee$ *(TestIssue(Issue))* $\bigwedge$ *isGenarates(ActResult,Issue)*

Further, *Test Issue Report*, could report this event in detail, which requires investigation. Finally, during a Test Result Analysis, *Test Results* are analyzed and a *Test Analysis Report* is produced.

B. Ontology Engineering Approach

Ontology engineering approach investigates the principles, methods and tools for initiating, developing and maintaining ontologies[14]. In literature, many methodologies have been proposed until now to build an ontology[15], [16], [17], but we considered the Grüninger and Fox methodology[18], which considered a formal approach to design ontology as well as providing a framework for evaluating the adequacy of the

developed ontology. This methodology focuses on building ontology based on first-order logic (FOL) by providing strong semantics. In our scenario, we introduce Description Logic(DL) which is a decidable fragment of FOL since we are designing with OWL 2 Web Ontology Language[19] for sematic web. The widely-used Protégé system (http://protege.stanford.edu) has recently been extended with support for the additional constructs provided by OWL2[20].

C. Contextual Information

We describe ”context specific”[21] to the software testers belonging to a leading software company in Sri Lanka and the approach which will be used to design the ontology to provide context-specific information and knowledge to software testers. To identify software tester's context clearly, we have extracted domain specific knowledge of software testing ontologies from existing literature and interviewed the software testing experts belonging to a particular company.

D. Competency Questions(CQs)

Competency Questions (CQ) are a set of questions that the ontology must be capable of answering using its axioms[18]. Thus, these CQs work as requirement's specification of the software testing ontology. With a set of CQs at hand, it is possible to know whether an ontology was created correctly, if it contains all the necessary and sufficient axioms that correctly answer the CQs. Some of the CQs used are shown in Table III.

TABLE I
TESTER'S INFORMATION NEEDS IN CONTEXT (I.E. COMPETENCY QUESTIONS)

Tester's Information Needs in Context (i.e. Competency Questions)
List out the human resources available in a testing activity?
Suitable test case design techniques for a given scenario?
What are the Test Levels described in software testing?
To which Test levels a particular TestDesignTechnique could be applied?
What kind of test automated software tools are available for use in your testing process?
Which Suitable Hardware Resources available for a given TestPlan?
What is the project name in which particular testing activity occurred?
What are the testing artifacts used in a testing activity?
What are the testing artifacts produced in a testing activity?
What are the testing objects used in a testing activity?

E. Ontology Components

At the first instance, high-level ontology concepts, their properties and their relationships should be identified. The basic high level ontology concepts are identified as Test Environment ,Testing Activities, Static Testing Techniques, Test Design Techniques, Test Objects, Test Level, Testing Artifacts and Organizational Team. For example, the concept Organizational Team having the properties of TeamID, TeamName, Team Size and Team Type. Secondly, sub classes of the high-level ontology concepts, their properties and relationships are also defined. For example, Test Environment has Human Resource, Software Tools and Hardware as it's sub classes. These sub classes are related to their superclass by is_ a relation. Artifact concept is mapped to disjoint and equivalent in OWL subclasses. During the modeling of software testing domain ontology, some special types of axioms such as Instantiation, Assertion, Subsumption, Domain, Range and Disjointness are included. The classes have been created in Protégé 5.1 can be described by necessary and sufficient conditions as illustrated in Table II.

TABLE II
NECESSARY AND SUFFICIENT CONDITIONS WRITTEN IN PROTÉGÉ 5.1

Testing Activities	Axioms written in Protégé 5.1 for Equivalent Classes
TestCaseDesign	(isGeneratesTestCase only TestCase) and (isGeneratesTestCase min 1 TestCase)
TestExecution	hasExecutesTestCases only (TestCase and (hasTestCaseExpectedResult some ExpectedResult))
TestPlanning	(hasCreatesTestPlan some TestPlan) and (hasCreatesTestPlan only TestPlan)
TestResultAnalysis	hasEvaluatesTestCases some (TestCase and (hasTestActualResult some ActualResult))

IV. DEVELOPMENT OF SOFTWARE TESTING KM PORTAL

As mentioned before, ontologies are powerful mechanism for representing knowledge presented in semantic web. The ontology and semantic web technologies provide powerful reasoning capabilities. In this section, we describe the ontology based knowledge management system to manage software testing knowledge, built upon the Java J2EE distributed component environment.

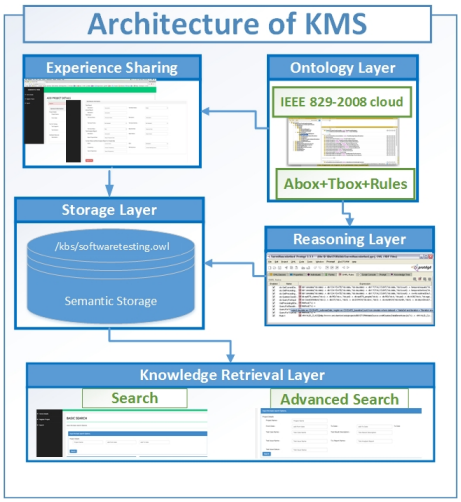


Fig. 1. Design of Knowledge Management Portal

Our KM Portal design consists of Experience Sharing, Ontology, Knowledge Retrieval, Storage and Reasoning Layer and is shown in Figure 1. The class diagram of Knowledge Sharing Portal is shown in Figure 2. We have implemented as a starting point of specifying the Knowledge Management Portal. Importantly, our presented concepts, properties and

relationships here are identified according to the characteristics of the particular organization's software testing environment.

A. Ontology Layer

The ontology layer has our ontology which includes domain rules, axioms etc. Using the Protégé Ontology Editor 5.1, these concepts and their relationships were partly described in section III.

B. Experience Sharing Layer

Through the Experience Sharing layer, software testers can annotate their testing knowledge with the support of ISTQB and IEEE 829-2008 terms. Once the knowledge is shared, such knowledge is transformed by the semantic data generator into the semantic data in a machine-understandable format of triple structure.

C. Storage Layer

We used Triple-store, which stores RDF triples and are queried using SPARQL. Jena TDB[22] has been selected to use in this study because it is a component of Jena for RDF storage and query. Importantly, it supports the full range of Jena APIs and can be used as a high performance of RDF store on a single machine.

D. Reasoning Layer

The Semantic Web Rule Language (SWRL) is based on a combination of OWL with the Rule Markup Language which provides inference capabilities[23] from existing OWL ontology. Rules in SWRL reason about OWL instances in terms of OWL classes and properties. Importantly, such rules express more complex relationships and restrictions between concepts. Software Testing rules were generated with Protégé SWRL Editor that is a plugin in Protégé environment and with the support of the Jess Rule Engine.

E. Knowledge Retrieval Layer

To show how our ontology can be used to share software testing knowledge collected from software testers, the Knowledge Retrieval Layer includes two functionalities that use Semantic Web technologies: (1) basic search, and (2) Advanced Search. SPARQL(SPARQL Protocol And RDF Query Language) has been used as the query language to retrieve software testing knowledge from the semantic data storage. The basic search provides a simple triple pattern matching service, which is one of the most frequently used functions for searching documents in the Semantic Web. Besides, Advanced Search Option includes, logical operators (AND or NOT or OR), so that user can combine different options to retrieve knowledge.

V. Ontology Evaluation

The quality of an ontology should be verified and validated before it is used in practice to avoid defects[24]. Further, such validation process will prevent contain ontology with anomalies or pitfalls, inconsistent incorrect and redundant information. Our evaluation of the ontology is carried out in three methods such as internal(using reasoners and OOPS!), ontology expert method and non-expert methods. Importantly, Protégé inbuilt reasoners such as FaCT++ 1.6.5 and HermiT 1.3.8.413 were used to check the internal consistency and inferences. OOPS!(http://oops.linkeddata.es/) is an online ontology evaluator has been used for our context to detect potential pitfalls that could lead to modelling errors, before ontology has been deployed in the end-user application. This method evaluates human understanding, logical consistency, modelling issues, ontology language specification, real world representation and semantic applications from the developed ontology[24]. Table III summarizes the pitfalls encountered, along with a brief description and the way each one of them was handled. There were three levels such as Critical, Important and Minor. Critical level is very crucial and it must be corrected in order to avoid ontology inconsistency. To make ontology nicer both Minor and Important cases were corrected.

TABLE III
Pitfall Description and solution Proposed

Pitfall	Description	Solution Proposed
Missing Annotation (256 cases | Minor)	Ontology terms lack annotation properties that could improve the understanding of the ontology	Included the ontology annotation properties
Missing domain or range in properties (7 cases |Important)	Relationships and (or) attributes without domain or range are included.	Restored missing domains.
Missing Inverse Relationships (14 cases| Minor)	When a relation has non-inverse relationship defined	Included Missing inverse relationships
Defining wrong inverse relationships. (6 cases | Critical)	Relationships are defined as inverse relations when they are not necessarily inverse.	They were removed to improve the expressiveness

Expert evaluation activity is performed by two ontology experts who have a good understanding of software testing. Even though there were many methods discussed to evaluate ontologies, our ontology experts considered Vocabulary, Syntax, Structure, Semantics, Representation and Context to perform evaluation[25]. The main objectives of expert evaluation are (a) whether the software testing ontology meets its requirements, standards, representation of concepts, relationships among them (b) coverage of the software testing domain and (c) checking for internal consistencies. The following suggestions and improvements were highlighted by the ontology experts: (a) Need of Annotations, renaming of concepts to standard methods. (b)Improving the relationship names, removing redundant relationships, some of the relationship were not mentioned, adding of association relationships which were missing. All such suggestions, comments were taken into consideration and implemented in the software testing ontology.

To carry out our first industrial application based evaluation from software testing experts, the developed Knowledge Management Portal was hosted locally inside one software company in Sri Lanka. A separate questionnaire was designed

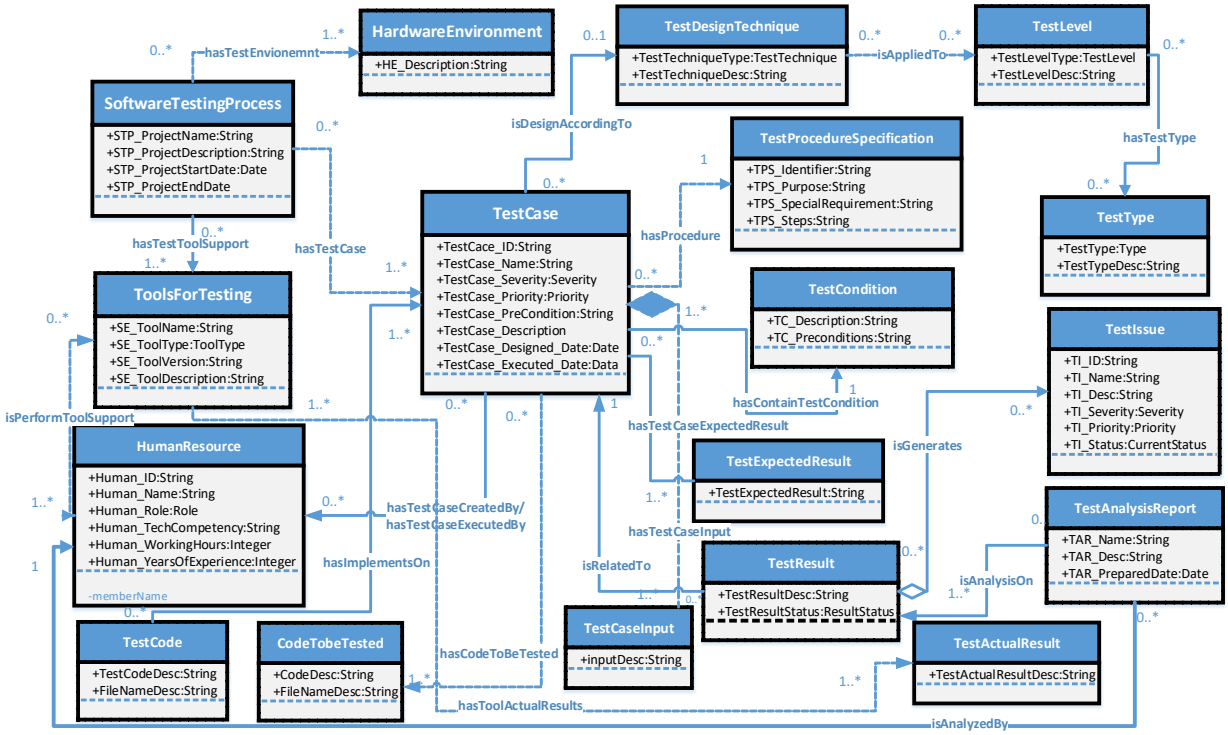


Fig. 2. Class Diagram of Knowledge Management Portal of Software Testing Knowledge

TABLE IV
SUMMARY OF THE SURVEY AND KEY QUESTIONS

Problem Identified from Survey - in the context of Software Testers (Beginning of the study)	**Feedback from the software testers (After developing Ontology and Implementing KM Portal)**
Is the knowledge in the knowledge repositories precise?	The large majority (66.67%) of the respondents from group A believed (Strongly Agree or Agree) that the provided content in Web Portal is precise. Only 44.44% of the respondents from group B believed (Strongly Agree or Agree) that knowledge in the knowledge repositories is precise.
To what extend is the software testing process incorporated?	Results illustrates that responded software testers have different views about the software testing process.
To what extend Internal documents are categorized using a standardized classification?	More than (75%) of software testers participated believed that the standards have been maintained
To what Extend Search or Retrieval Functionalities of developed KM portal support software testing?	Group A (41.66%) of the respondents believed (Strongly Agree or Agree) that the search or retrieval functionalities are adequate while from group B this value is 55.56%.
To what Extend Knowledge Sharing Functionalities of developed KM portal support software testing?	41.66% respondents from group A believed (Strongly Agree or Agree) that the knowledge sharing functionalities are adequate. 66.67% of the respondents from group B agreed(Agree) with the existing KM portal.
To what extend do you think it useful applying KM Practices in Software Testing?	Both groups (A- 41.67%, B-55.56%) believed with such practices and such results leads to conclude at least the importance and the need of KM practices for their daily activities.

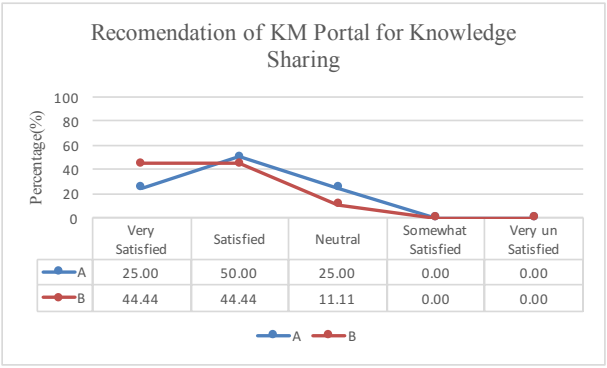


Fig. 3. Recommendation to use Ontoloy-based KM Portal

in English with five-point Likert-type scale to capture respondents' self-reported attitudes where respondents had to make their level of agreement such as; Strongly Agree, Agree, No Idea, Disagree and Strongly Disagree. Scores 5, 4, 3, 2, and 1 were assigned respectively for the above-mentioned categories. The profiles and demographics of the participants (Employed Group, work experience, job description, and qualification) were questioned first and continued with questions focused to check whether developed ontology was able to (a) express software testing knowledge (b) support software testing knowledge sharing (c) support software testing knowledge retrieval and (d)User Satisfaction. We limited the time period to ten (10) days to collect the questionnaire data. This was performed by

two software testing groups working on a similar information system project for comparison and in depth understanding. Importantly, a prior training session was conducted through recorded video to make software testers familiar with the Knowledge Management Portal.

The results are summarized in Table IV. When asked to evaluate the content of the KM Portal, Group B believes that correct contents have been included than the Group A. All of the software testers participated believed that the standards have been maintained, but a few have included some extra comments, such as inclusion of some vocabularies, parameters in Test Level and Test Types. Moreover, when asked about their satisfaction on the factors related to user manipulation, personalization and knowledge community, both groups have mostly responded neither satisfied nor unsatisfied. Notably, a very few have been very much satisfied with the user manipulation or knowledge community of the KM Portal. To conclude, overall 80.95% (See Figure 3) of the respondents would like to recommend the use of such KM Portal among the software testers for knowledge sharing and knowledge retrieval.

VI. Discussion and Conclusion

This research presents software testing ontology to represent software testing domain knowledge which includes software testing concepts, properties and their relationships. The implemented an ontology-based KMS based on semantic web technologies provides facilities for software testers to share their knowledge and experiences. Basic search and advanced search operations are used to retrieve knowledge. The implemented ontology was validated and evaluated before it is used. Ontology experts opinion was received to improve the ontology. The results from the industrial experimental investigation show that the proposed ontology-based KM Portal is adequate to support knowledge sharing and reuse, allowing: (a) knowledge representation and organization; (b) distributed knowledge inference and retrieval; (c) management of organizational knowledge on software testing. Our Portal addresses the existing key issues such as knowledge is not reaching the software testers due to its unstructured, incomplete, varied formats, and lack of targeted delivery methods. We believe our software testing ontology can support other software organizations to improve the sharing of knowledge and learning practices. In the future work, the reasoning engine with Query-enhanced Web Rule Language (SQWRL) will be incorporated into knowledge searching to support more precise and effective knowledge sharing.

Acknowledgment

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions. This work was partially supported by the National Natural Science Foundation of China (Grant No. 61672398), the Key Natural Science Foundation of Hubei Province of China (Grant No. 2015CFA069), and the Applied Foundamental Research of Wuhan (Grant No. 20160101010004).

References

[1] I. Rus and M. Lindvall, "Knowledge management in software engineering," *IEEE software*, vol. 19, no. 3, pp. 26–35, 2002.

[2] S. Vasanthapriyan, J. Tian, and J. Xiang, "A survey on knowledge management in software engineering," in *Software Quality, Reliability and Security-Companion (QRS-C), 2015 IEEE International Conference on*. IEEE, 2015, pp. 237–244.

[3] V. Santos, A. Goldman, and C. R. De Souza, "Fostering effective inter-team knowledge sharing in Agile software development," *Empirical Software Engineering*, vol. 20, no. 4, pp. 1006–1051, 2015.

[4] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?" *International journal of human-computer studies*, vol. 43, no. 5-6, pp. 907–928, 1995.

[5] I. Horrocks, "Ontologies and the semantic web," *Communications of the ACM*, vol. 51, no. 12, pp. 58–67, 2008.

[6] S. S. E. Committee *et al.*, "IEEE standard for software and system test documentation," *Fredericksburg, VA, USA: IEEE Computer Society*, 2008.

[7] D. Graham, E. Van Veenendaal, and I. Evans, *Foundations of software testing: ISTQB certification*. Cengage Learning EMEA, 2008.

[8] O. K. Wei and T. M. Ying, "Knowledge management approach in mobile software system testing," in *Industrial Engineering and Engineering Management, 2007 IEEE International Conference on*. IEEE, 2007, pp. 2120–2123.

[9] X. Li and W. Zhang, "Ontology-based testing platform for reusing," in *Internet Computing for Science and Engineering (ICICSE), 2012 Sixth International Conference on*. IEEE, 2012, pp. 86–89.

[10] I. Douglas, "Testing object management (TOM): A prototype for usability knowledge management in global software," in *International Conference on Usability and Internationalization*. Springer, 2007, pp. 297–305.

[11] E. F. Souza, R. Falbo, and N. L. Vijaykumar, "Using ontology patterns for building a reference software testing ontology," in *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2013 17th IEEE International*. IEEE, 2013, pp. 21–30.

[12] N. Bajnaid, R. Benlamri, A. Pakstas, and S. Salekzamankhani, "Software quality assurance ontology from development to evaluation (s)." in *SEKE*, 2013, pp. 689–694.

[13] G. J. Myers, C. Sandler, and T. Badgett, *The art of software testing*. John Wiley & Sons, 2011.

[14] Y. Sure, S. Staab, and R. Studer, "Ontology engineering methodology," in *Handbook on ontologies*. Springer, 2009, pp. 135–152.

[15] M. Fernández-López, A. Gómez-Pérez, and N. Juristo, "Methontology: from ontological art towards ontological engineering," 1997.

[16] Y. Sure, S. Staab, and R. Studer, "On-to-knowledge methodology (otkm)," in *Handbook on ontologies*. Springer, 2004, pp. 117–132.

[17] N. F. Noy, D. L. McGuinness *et al.*, "Ontology development 101: A guide to creating your first ontology," 2001.

[18] M. Grüninger and M. S. Fox, "Methodology for the design and evaluation of ontologies," 1995.

[19] W3C OWL Working Group, *OWL2 Web Ontology Language: Document Overview*. W3C Recommendation, 2009, available at http://www.w3.org/TR/owl2-overview/.

[20] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler, "OWL2: The next step for OWL," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 6, no. 4, pp. 309–322, 2008.

[21] A. K. Dey, "Understanding and using context," *Personal and ubiquitous computing*, vol. 5, no. 1, pp. 4–7, 2001.

[22] K. Wilkinson, C. Sayers, H. Kuno, and D. Reynolds, "Efficient RDF storage and retrieval in Jena2," in *Proceedings of the First International Conference on Semantic Web and Databases*. CEUR-WS. org, 2003, pp. 120–139.

[23] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean *et al.*, "SWRL: A semantic web rule language combining OWL and RuleML," *W3C Member submission*, vol. 21, p. 79, 2004.

[24] M. Poveda-Villalón, M. C. Suárez-Figueroa, and A. Gómez-Pérez, "Validating ontologies with OOPS!" in *International Conference on Knowledge Engineering and Knowledge Management*. Springer, 2012, pp. 267–281.

[25] D. Vrandečić, "Ontology evaluation," in *Handbook on Ontologies*. Springer, 2009, pp. 293–313.

A Framework for Developing Cyber Physical Systems

Xudong He
Florida International University, Miami, USA
Heng Yin
University of California at Riverside, Riverside, 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 framework to develop CPSs based on a model driven approach with quality assurance throughout the development process. An agent-oriented approach is used to model individual physical and computation processes using high level Petri nets, and an aspect-oriented approach is used to integrate individual models. The Petri net models are systematically mapped to classes and threads in Java, which are enhanced and extended with domain specific functionalities. Complementary quality assurance techniques are applied throughout system development and deployment, including simulation and model checking of design models, model checking of Java code, and run-time verification of Java executable. We demonstrate our framework using a car parking system.**

Keywords - cyber physical systems; model driven development; 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, incremental improvements of existing technologies including formal verification, simulation, software engineering processes, and design patterns are important parts of a potential solution [2].

In this paper, we provide a concrete framework to realize the ideas in [2]. We present a model driven approach from high level Petri nets to Java programs where several design heuristics are provided for program derivation and system properties mapping. 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.

Petri nets are a formal method well suited for modeling concurrent and distributed systems. Various time extended Petri nets are capable to deal with real-time systems [3]. High level Petri nets can use time stamps associated with tokens and timing related transition constraints to simulate time Petri nets [4]. Thus high level Petri nets are an excellent formal method for

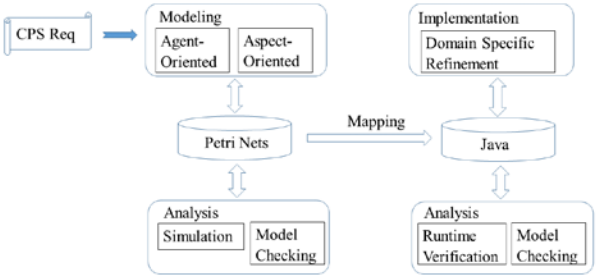


Fig. 1 – A framework for developing cyber physical systems

modeling essential features of CPSs. In addition, we have developed an agent oriented modeling approach to capture CPSs at a high abstraction level where meaningful computational components and physical processes with independent behaviors are viewed as agents and modeled individual high level Petri nets. An aspect oriented approach is used to incrementally integrate system components represented using individual high level Petri nets into a complete system represented in a single system high level Petri net. The resulting system net can be analyzed through simulation as well as model checking. The above modeling and analysis techniques are supported by tool chain PIPE+ [5] and SPIN [6]. 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.

Our main contributions include: (1) a formal framework for developing CPSs supported by a tool chain, (2) an incremental agent-oriented modeling methodology for representing CPSs using high level Petri nets, (3) a model level analysis methodology combining simulation and model checking, (4) a pattern based translation method for generating Java threads from high level Petri net models, and (5) an implementation level analysis methodology combining bounded symbolic model checking and dynamic runtime verification.

II. CPS MODELING AND ANALYSIS

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 [7] and special graphical modeling languages such as actor-oriented MoC [8]. High level Petri nets [9] are well suited to model the complex behaviors of CPSs. The graphical representation and data flow nature of Petri nets provide a natural and easy to understand model to capture physical and computation processes in CPSs. The executability of Petri nets further facilitate model level analysis. In this paper, we propose an agent-oriented approach to model individual physical and computation processes by extending our prior work [10] and an aspect-oriented approach [9] to synthesize individual models to obtain a complete system model. In the following sections, we provide some design heuristics of applying high level Petri nets to model CPSs and demonstrate them using a car parking system. The detailed Petri net definitions are omitted due to space limit.

A. 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 a high level Petri net called an agent net that has its own meaningful and demonstrates independent reactive and/or proactive agent behavior interacting with external environments, while concrete physical devices or computation processes are with structured tokens containing unique identification in the 1st field. Specifically, we provide the following general design heuristics in building an agent net:

- Attributes of a physical device or states of a computation process are defined by places with appropriate types. Discrete values are defined using string or integer types and continuous values are defined using real type. Structured types (Cartesian product of basic types) are used to define complex attributes. Powerset is used to define multiple physical devices and computation processes;
- Actions or state transitions are modeled with transitions containing first order logic formulas defining the preconditions and post-conditions;
- The interaction between a physical device and an external environment can be modeled with a transition containing a random function emulating the possible values from the environment (open system) or with a transition picking up a possible value from an additional place denoting the external environment (closed system);
- Virtual time is modeled with tokens having an additional field denoting time stamps and a special place modeling a logical clock.

We demonstrate the above design heuristics in modeling a robotic car parking system. Each robotic car has one color sensor for navigating a path with colored lines and an ultrasonic sensor for detecting obstacle during parking. The car parking process involves the following steps: (1) finds the entrance of a parking garage by detecting a green line, (2) moves forward along a red line, (3) makes a turn when a blue line is detected, and (4) completes the parking when the minimum specified distance is reached. Based on the above simple system description. Three individual agent nets corresponding to the color sensor, the ultrasonic sensor, and the car parking process are constructed. Only essential attributes of the sensors and car are represented, for example, only one place *ColorSensor* for holding the current detected color is needed for the color sensor.

B. Modeling the Whole System

The overall agent system is obtained by integrating individual agent nets to form a system net 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 [9] 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. The overall car parking system model after weaving three agent nets is shown in the following Fig. 2.

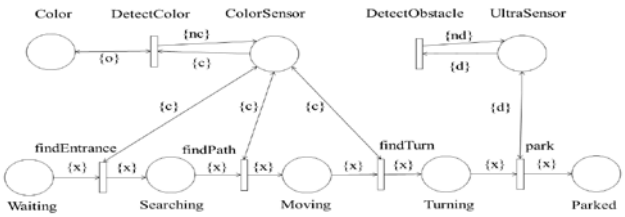


Fig. 2 – The overall system integrating three agent models

C. Analyzing the System Model

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. However, 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. High level Petri nets are executable and thus support simulation of hybrid system models. Furthermore our tool PIPE+ supports reachability analysis and model checking using SPIN in addition to simulation.

1) Simulation and Simple Reachability Analysis

Simulation is carried by firing enabled transitions. Two modes of simulation can be done in PIPE+: single steps and multiple steps. Simulation can be used to test whether a model

satisfies the CPS requirements by examining the tokens in places of interest or transition firing history. In addition to simulation, we have also implemented a simple reachability analysis that checks whether a place hold a particular value during an execution, which either confirms our intention (witness) or finds a potential error (counter example). When a reachability is confirmed, the simulation time and the transition firing sequence are recorded. However simple reachability analysis may not be conclusive when a search does not end. Repeated checking can be used to eliminate possible false negative in systems with finite execution sequences.

The following table provides a simple reachability analysis of whether a waiting car will be parked.

Table I – Simple reachability checking results

Parked	Reachability	Time (ms)
c	Yes	459

The firing sequence is very long in this case since both the color sensor and the ultrasonic sensor are modeled as independent agents, which are always enabled and keep firing.

2) Model Checking

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.

Properties to be checked are expressed using linear time temporal logic formulas. Safety and liveness properties are expressed in the general form []*placename*(*x*) and <> *placename*(*x*) respectively, where [] and <> 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. A safety property of the parking system is that a car is collision free during parking []!(*UltraSensor*(*v*)&&*v*<5), where s is the ultrasonic sensor detected distance value. In the model, we can only use assumed value range for checking. A liveness property of the car parking system is that a waiting car will eventually be parked. Table II provides model checking results of the above properties.

Table II – Model checking results

Property	Satisfied	Time (ms)
[]!(*UltraSensor*(*v*) && *v* <5)	Yes	47
Waiting(*c*)-><>*Parked*(*c*)	Yes	1

III. 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 [11], 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. We present a model driven approach to realize our high level Petri net models, which provides 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 can be 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.

The following translation rules are used to generate the general code structure from high level Petri net models:

(1) A class is generated for each agent net, where attributes are defined based on the unique data type fields of the places, and methods are the transitions. The behavior of objects (tokens) is defined by the net structure;

(2) A thread is created based on each class in (1) to capture the independent active behavior of agents modeled by the agent net;

(3) Agent interactions modeled through agent net weaving are translated into method calls between threads in (2);

(4) A main program is generated for the overall model, which starts all the threads generated in (2) according to the initial marking;

(5) A package is created to include the above code files.

Applying the above translation rules to the car parking system, we obtain the following Java code skeleton (due to space limit, only the translation of car parking agent model and overall system model are shown) together with some blue colored domain specific refinement code:

- The `Car.java` class and the `Parking.java` thread from parking process agent model:

```
package EV3;
import lejos.hardware.motor.Motor;
import lejos.hardware.motor.NXTRegulatedMotor;
…
public class Car {
    float wheelWidth = 5.5; // in cm
    float trackWidth = 30.0; // in cm
    NXTRegulatedMotor leftM = Motor.A;
    NXTRegulatedMotor rightM = Motor.B;
    UltraSensor ultrasensor;
    ColorSensor colorsensor;
    …
}
package EV3;
…
public class Parking extends Thread {
    private Car carobj;
    public void findEntrance (…) {…}
    public void findPath (…) {…}
    public void findTurn (…) {…}
    public void park (…) {…}
    …
    public void run {
```

```
                findEntrance(…);
                findPath(…);
                findTurn(…);
                park(…);
        }
    }
```

- The main program from the system model:

```
package EV3;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import lejos.robotics.Color;
import lejos.utility.Delay;
public Class Init {
    public static void main(String[] args){
     Car car = new Car();
     ColorDetection cd = new ColorDetection(car);
ObstacleDetection od = new ObstacleDetection(car);
     Parking p = new Parking(car);
     cd.start();
     od.start();
     p.run();
   }
}
```

In the above code, `ColorDetection` is a thread object that asks the color sensor to keep reading and storing the current color value as long as parking is not completed. The color sensor is represented by the class `ColorSensor`. `ObstacleDetection` is also a thread object that asks the ultrasonic sensor to keep reading the distance to the object ahead of the car in current path as long as parking is not completed. The ultrasonic sensor is represented by the class `UltrasonicSensor`.

IV. Runtime Verification

A good model can speed up the development of the system and provide quality assurance to the implementation. However, it doesn't guarantee the correctness of the implementation due to several reasons. First, the model is an abstraction of the system. Some details and algorithms are omitted intentionally and some system properties cannot be specified. Second, the model driven approach requires some manual refinements and the complete system with additional code needs to be checked.

To overcome the limitations and restrictions of model driven development, we adopt runtime verification to ensure system properties at implementation level. Runtime verification is a lightweight formal approach to detect violation of properties. In our work, properties are specified using linear temporal logic (LTL) formula. Monitors are generated from these LTL formulas using JavaMop [12] and woven into system implementation as aspects using AspectJ [13]. This ensures the independence of system implementation from monitor – the runtime verification code.

LTL properties specified and analyzed at model level need to be monitored in implementation level to improve confidence of system implementation. However, atomic predicates of LTL formula in implementation level typically represent occurrences of events such as object creation, object initialization, method call, member data access and mutation. Such events can be specified in conjunction with conditions on function arguments, which are absent at model level. The following principles are provided to guide the mapping of atomic predicates (places) in Petri net model to implementation:

(1) A place representing a state of the agent net:

a. If a class generated from the agent net has a member data representing the state, then the place is mapped to the event of changing member data. For example, place Parked represents a state of car and class `Car` has a member data `parkComplete` denoting this state, thus Parked(c) is mapped to the event of `Car.parkComplete` being set to true.

b. If a class generated from the agent net doesn't have a member data representing the state, then an object reaching this state is a result of a function call and is mapped to the event that occurs whenever the function call is completed.

(2) A place representing the duration of an action such as Turning in the agent net: the place is mapped to the event of executing the function starting the action.

In our study, five different properties are designed and monitored at runtime, including:

Property 1: After detecting the entrance, a car will park successfully within a given time period;

Property 2: A car starts turning to a parking lot only after a blue line is detected;

Property 3: During the parking process, a car will never approach too close to any object on the path.

Due to space limit, we only provide and discuss the monitor code of property 1. This liveness property is to guarantee the correctness of the overall program: A car eventually parks successfully. In the model level, it is specified as: [](*Waiting*(*c*) -> <>*Parked*(*c*)). This property can be verified in the model but its violation cannot be determined at runtime due to its infinite nature. Therefore, we modify the property to reflect the reality – the car must be parked within a given time period. Such modification is reasonable given the unreliable nature of robots: the color sensor may report wrong color or miss a color. The monitor of this property contains the following three main events: `entranceDetectedtrue`, `parkCompletedtrue` and timeout. Event `entranceDetectedtrue` occurs when car's member data `entranceDetected` becomes true. When `entranceDetectedtrue` occurs, a timer is started. Event `parkCompletedtrue` occurs when car's member data `parkComplete` is set to true. Event timeout occurs when a given time expires.

```
event entranceDetectedtrue after(Car car, boolean val) :
     set(private boolean Car.entranceDetected)
     && args(val)  && target(car) && condition(val){
        //start timer
        …
     }
event parkCompletedtrue after(Car car, boolean val) :
     set(private boolean Car.parkComplete)
     && args(val) && target(car) && condition(val)) {
        //kill timer
     }
event timeout after(TimeoutEventGenerator t) :
execution(publicvoidTimeoutEventGenerator.timeoutEvent())
     && target(t) && condition(t == teg) {
     }
ltl:[](entranceDetectedtrue =>
           (!timeout U parkCompletedtrue))
```

The second property in model level specified as: [](*Searching*(*c*) => <>*ColorSensor*(*Blue*)) can be verified at both model level and implementation level.

The third property is a safety property to ensure collision

free during parking. In model level, it is specified as: [](*UltraSensor*(*v*) /\ *v* < value), where value is a given constant. However, the verification results of such safety properties are based on some assumptions, and thus their violations must be monitored during runtime.

Therefore runtime verification complements model level and implementation analysis through monitoring program behaviors that cannot be fully verified at model and implementation levels due to the limitation of models or bounded space explosion at implementation level.

V. Implementation-Level Model Checking

Implementation level model checking is complementary to both modeling and runtime verification. While modeling is concerned about the correctness at design level, as the name suggests, implementation level model checking aims to verify the properties in the actual software implementation, because errors may be introduced during the software implementation.

It is also well accepted that model checking and runtime verification should be combined to address each other's weaknesses [14, 15]. Model checking achieves completeness, but often is too conservative and impractical for many realistic applications, whereas runtime verification can perform excellent checks over a portion of the program that is actually executed during the real deployment.

Researchers have explored different approaches to combine model checking and runtime verification. For instance, [15] presents an analysis framework that can reuse the same analysis/verification algorithms for both static and dynamic analysis, in other words, model checking and runtime verification. While this is useful, it does not answer when to use model checking and when to use runtime verification. [14] explores the idea of partitioning a software system, such that one partition can be verified using model checking and the other can be checked via runtime verification. However, a user must identify this partitioning boundary based on her domain knowledge, which may not be practical in reality.

A. Bounded Symbolic Model Checking

In this work, we explore a new approach to combine model checking and runtime verification, namely bounded symbolic model checking. Specifically, we perform symbolic execution to examine the program execution space as much as possible to directly verify the property validation logic inserted by runtime verification. Given that a program may have infinite loops or loop conditions depending on symbolic inputs, the program execution space is infinitely large. As a result, our symbolic execution is bounded to search limited iterations in each loop.

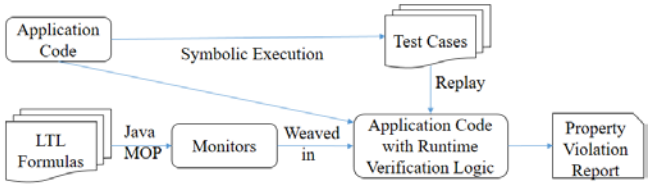


Fig. 3 – Workflow of Bounded Symbolic Model Checking

Fig.3 illustrates the workflow of this technique. Given a CPS program, we perform symbolic execution to generate a set of test cases, which will exercise as many code paths as possible. These test cases are then fed into the program with runtime verification code weaved in, to verify the properties.

It is worth noting that we do not directly perform symbolic execution on the program with runtime verification code weaved in. This is because the inserted runtime verification code greatly increases the complexity of the code logic, and symbolic execution on it will quickly run into state explosion problem.

Moreover, for convenience, we directly take advantage of the runtime verification code that has already been inserted, rather than verify the properties separately. In this way, we can also detect errors introduced by runtime verification.

B. Implementation

We implemented this idea as a plugin to the Java Path Finder framework [16] to perform model checking on LeJOS programs

1) JPF model classes

A LeJOS program can only run inside the LeJOS environment. As a result, it cannot run directly within JPF. In order to analyze a LeJOS program within JPF, we need to provide proper model classes to simulate the behaviors of related LeJOS APIs and classes. More specifically, we model following LeJOS classes:

```
lejos.hardware.lcd.LCD;
lejos.hardware.motor.BaseRegulatedMotor;
lejos.hardware.sensor.EV3ColorSensor;
lejos.hardware.sensor.EV3UltrasonicSensor;
lejos.hardware.sensor.SensorMode;
lejos.hardware.sensor.UARTSensor;
lejos.robotics.navigation.MovePilot;
```

2) JPF Symbolic Execution

We make use of JPF-symbc to perform symbolic execution. For the auto parking example, the program reads input from the color sensor and the ultrasonic sensor. Therefore, by adding JPF symbolic annotations, we make these sensors return symbolic values.

Since the auto parking program repeatedly reads from the sensors and perform corresponding behaviors, there exist several infinite loops. To ensure symbolic execution terminates within a reasonable time frame, we limit the number of loop iterations, by using JPF Verify API.

Following is a code snippet showing how we limit loop iterations in the auto parking program:

```
int in_count = 0;
int out_count = 0;
while (carObj.isFindLine()) {
    int color = getColorSin(1);
    while(color != Color.RED && color != Color.BLUE){
        in_count ++;
        Verify.ignoreIf(in_count > 3);
        ...
    }
    out_count++;
    Verify.ignoreIf(out_count > 2);
    ...
}
```

In this example, there are two while loops, inner loop and

outer loop. There is a chance of program being trapped in either inner loop or outer loop (or both) during execution. So, we add a counter for each loop. When the counter is larger than a threshold, JPF will not continue execute the current path. We then solve the constraint for each path, and generate test inputs.

C. Evaluation results

Table III – Bounded symbolic model checking results

# of test cases	# of violations	Generation time	Replay time
688	344	519 s	13120 s

Table III lists the performance results for our bounded symbolic model checking on the original auto parking example. Symbolic execution generated 688 test cases totally in 519 seconds.

We then replayed these test cases on the auto parking program with runtime verification logic weaved in. The replay lasted for 13120 seconds. This process is significantly slower, because the auto parking program periodically calls `Delay.msDelay(int time)` to wait for amount of time. Symbolic execution would ignore this delay to quickly explore multiple execution paths, while during replay in order to validate time-related properties, we cannot ignore these delays.

Eventually, we observed violations in 344 test cases. All the violations happened on only one LTL property, which is collision free property. This property is violated when the EV3 car approaches too close to an obstacle.

```
public void turn() {
    if (carObj.getCMD() == 1){
        carObj.pilot.stop();
        carObj.pilot.setAngularSpeed(7);
        carObj.pilot.arc(25, 90, true);
        carObj.setMoving(true);
        Delay.msDelay(15000);
    }else if (carObj.getCMD()==0){
        carObj.pilot.stop();
        carObj.setMoving(false);
    }
}
```

In the above example, the EV3 car would possibly turn for 15000 milliseconds. After each turn, Ultrasonic sensor is used to detect distance between an obstacle and EV3. When the EV3 car is running in a physical system (parking lot), this could be a safe action due to parking lot layout. However, during symbolic execution without knowledge about physical system, JPF explores all possible paths, and could read in a distance value less than safe distance after such a long turn, and results in a violation of collision free property.

Through bounded symbolic model checking, we have verified that four LTL properties are ensured at least within a limited search scope. When resource is limited on the LeJOS system, one might consider removing the runtime monitor code for checking these four properties. On the other hand, collision free property cannot be properly verified using model checking, so runtime verification on this property is necessary.

VI. Conclusion

This paper presented a framework 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. An incremental agent-oriented modeling methodology is used for creating CPS models. The resulting models are analyzed using simulation and model checking to detect early design problems. A translation method for generating general Java thread structure from high level Petri net models is provided. The resulting general Java code structure is manually extended with domain specific code refinement 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. Implementation level quality assurance is carried out by combining bounded symbolic model checking and dynamic runtime verification. We demonstrated our framework thorough a simple parking system. There are still many gaps to fill to make our framework successful. We are currently working on a multi-car parking system and 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] D. Xu, X. He, and Y. Deng: “Schedulability Analysis of Real-Time Systems Using Time Petri Nets”, IEEE Transaction on Software Engineering, vol.28, no.10, 2002, 984-996.

[4] C. Ghezzi, D. Mandrioli, S. Morasca, and M. Pezzi: “A Unified High-Level Petri Net Formalism for Time-Critical Systems”, IEEE Transactions on Software Engineering, vol.17, no. 2, 1991, 160-172.

[5] Su Liu and Xudong He: “PIPE+Verifier - A Tool for Analyzing High Level Petri Nets”, Proc. of the 27th International Conference on Software Engineering and Knowledge Engineering (SEKE15), Pittsburgh, July 6 – 8, 2015.

[6] Gerard Holzmann: The SPIN Model Checker, Addison Wesley, 2004.

[7] 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.

[8] P. Derler, E. Lee, and A. Vincentelli: “Modeling Cyber-Physical Systems”, Proceedings of the IEE, vol. 100, no.1, 2012, 13 – 28.

[9] 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.

[10] L. Chang, S. Shatz, and X. He: “A Methodology for Modeling Multi-agent Systems using Nested Petri Nets”, International Journal of Software Engineering and Knowledge Engineering, vol.22, no.7, 2012, 891-926.

[11] B. Selic: “The Pragmatics of Model-Driven Development”, IEEE Software, 2003, 10 – 25.

[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] T. Hinrichs, P. Sistla, and L. Zuck: “Model Check What You Can, Runtime Verify the Rest”, EPiC Series in Computing, vol. 42, 2014, 234 – 244.

[15] C. Artho and A. Biere: “Combining Static and Dynamic Analysis”, Electronic Notes in Theoretical Computer Science, vol.131, 2005, 3 – 14.

[16] Java Path Finder: http://javapathfinder.sourceforge.net.

Towards Code Generation from Design Models

Pengyi Li*, Jing Sun†
*Department of Electrical and Computer Engineering
†Department of Computer Science
The University of Auckland, New Zealand
Emails: *pli552@aucklanduni.ac.nz, †j.sun@cs.auckland.ac.nz

Hai Wang
School of Engineering and Applied Science
Aston University, Birmingham B47ET
United Kingdom
Email: h.wang10@aston.ac.uk

***Abstract*—With the growing in size and complexity of modern computer systems, the need for improving the quality at all stages of software development has become a critical issue. The current software production has been largely depended on manual code development. Despite the slow development process, the errors introduced by the programmers contribute to a substantial portion of defects in the final software product. This paper explores the possibility of generating code and assertion constraints from formal design models and use them to verify the implementation. We translate Z formal models into their OCL counter-parts and Java assertions. With the help of existing tools, we demonstrate various checking at different levels to enhance correctness.**

I. INTRODUCTION

With the growing in size and complexity of modern computer systems, the need for improving the quality at all stages of software development has become a critical issue [1]. The focus lies in how to effectively develop high quality software. Looking at a topical software development life cycle, there are two main types of potential errors to a software product, i.e., design errors and programming errors [3], [4]. The former refers to the defects introduced at the design stage where the proposed product failed to capture the correct functionalities of the system under construction. The latter refers to the defects introduced at the implementation stage where the programmers failed follow the correct design in producing the program (code).

The key issue is how to effectively and efficiently develop code from verified design models [7]. Automation reflects an essential motive and aspect of future software engineering practice. The era of computing has revolutionised the mathematical computation and data processing from human-power into machine-power. However, the current software construction is still largely depended on manual code development, i.e., humans write programs manually from the design solutions. Despite the slow development process, the errors introduced by the programmers contribute to a substantial portion of defects in the final software product. Ideally, the executable program (code) should be automatically generated from the design model [5], which has been known in other engineering disciplines, e.g., mechanical and electrical, as the production automation. This will not only dramatically increase the productivity, but also overcome the human errors at the implementation phase and hence improve the overall quality of modern software development.

DOI reference number: 10.18293/SEKE2017-133

Currently, there are few theories being developed to implement the automatic code generation. Most related works are based on diagrammatic notations and generating partial code framework, such as UML to Java [2], [11]. Advantages of such an approach is that UML is nowadays the most commonly used notation for documenting design models and therefore this approach could become a widely acceptable and practical way in industry. However, this traditional model language is contributing relatively little to productivity [4]. Despite the informal description of the design notations, the generated programs are skeleton code only, which are not executable. Refinement [12] is another related field that aims at deriving less abstract models from formal specifications. However, to the best of our knowledge, there has been no work in refining a formal model directly into executable programs.

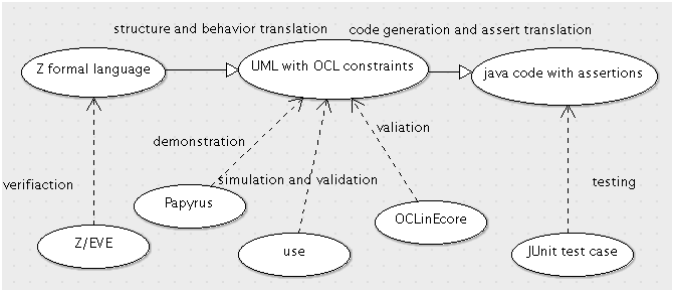


Fig. 1. Overview of our approach

In this paper, we explore the possibility of generating code and assertion constraints from formal design models and use them to verify the implementation. We translate Z formal models into their OCL [8] counter-parts and Java assertions. With the help of existing tools, we demonstrate various checking at different levels to enhance correctness. Figure 1 illustrates a summary of our approach.

Firstly, the design models are documented using the Z formal specification language [10]. Z is a first order logic and set theory based notation, which has a number of tools in supporting the verification and validation of on the design, such as Z/EVES, CZT, ProB, etc. Due to the high level abstraction of the Z language and its lacks of object oriented design, an

intermediate process is necessary for code generation. On the other hand, UML, a semi-formal diagrammatic notation, is a widely used design language by the industry and could well serve the purpose. In our project, Java is selected as the target implementation, and there are a number of tools support the generations of skeleton code from UML to java.

Secondly, to overcome the informal descriptions of UML, the formal specification from the Z model can be translated into OCL annotations. According to the steps shown, the approach translates Z formal models to UML with OCL constraints and then generate code with embedded assertions from UML to Java. Thirdly, to further guarantee correctness, OCL constraints are translated to Java assertions to validate the implementation. Furthermore, OCL constraints can be used in JUnit [6] test cases, which is easier for the developers to debug their code.

Finally, in addition to code generation, there are tools that could be used to support the rigorous verification at different stages of the development. For example, Z/EVE provides theorem proving to verify the desired properties at the specification level. At the UML/OCL phase, Papyrus can interpret UML models with OCL constraints and generate Java code. OCLinEcore and USE both provide simulation facilities for creating object models and validating OCL constraints. At the code level, Java assertions can be used for validating the implementation. And assertions can also be used together with JUnit for testing purposes.

The rest of the paper is structured as follows. Section 2 introduces the model transformation from Z to UML/OCL. Section 3 presents the translation from UML/OCL to Java code. Section 4 describes the benefit of translations in terms of verification, simulation, code generation and validation. Section 5 concludes the paper and outlooks the future work.

II. Translating Z into UML/OCL

This section describes the translation from Z formal models into UML/OCL specifications. The mapping consists of two parts, i.e., structure translation and behaviour translation.

A. Structure Translation

1) Z Schema: A Z formal specification contains state schemas and operation schemas. The state schemas can be translated to classes in UML/OCL. The operation schemas can be translated into methods belongs to specific classes. Operation schemas that change the value of state variables are indicated using the "Δ" convention; operation schemas that do not change state variables are indicated with the "Ξ". The state schema name that after the "Δ"or "Ξ" symbol can be used to identify the class that a method is belonging to.

2) Z Data Types: A Z schema consists of two parts, i.e., declarations and predicates. The declaration part of structure translation can be summarise into data type mappings to their counterparts in UML/OCL.

a) Given, basic and free types: Z uses [] to express an new given type. For example, [MSG] in Z can be translated to a class MSG with no attributes. Z only supports two primitive data types, i.e., nature numbers and integers, which can be mapped to integers in OCL with additional constraint, such as 'n>0'. Free types in Z can be mapped into enumeration types with values in OCL. For example, 'Z: StockStatus::=InStock | OutofStock' would be mapped into 'enum StockStatus {InStock,OutofStock}' in UML/OCL.

b) Sets: are expressed as {x:X | P(x)} in Z, which can be directly mapped into the Set type in OCL. In Z formal language, S$\subseteq$T means that S is a subset of T, which is equivalent to T$\rightarrow$includesAll(S); For example, 'products:P SaleItem' in Z can be translated into 'products: Set(SaleItem)' in UML/OCL. Cardinality #X in Z is a natural number denoting the size of a set. In UML/OCL, there is a predefined operation $\rightarrow$size() that has same meaning with "#".

c) Cartesian Product, Relations and Functions: : A$\times$B (A cross B) in Z can be translated in to an association class between two participating classes in UML/OCL with multiplicity constraint as many to many. A relation R from A to B in Z represents a subset of the cartesian product with special restrictions. This can be translated into invariant definitions on the associate class to restrict the mapping. A function R: A$\rightarrow$B in Z differs from a relation in the number of valid mappings through the relationship, i.e., for each a $\in$ A there is at most one b $\in$ B. In UML/OCL functions be denoted as an association class that connects class A with class B, and this association has multiplicity of many to one.

d) Sequences: in Z represents a sequence of elements from a set A, denoted as 's:seq A'. UML/OCL also has a concept 'Seq' to express a sequence of instances. Therefore, a direct mapping from 'seq' in Z to the 'Seq' type in UML/OCL.

B. Behavioural Translation

The syntax of standard OCL language uses contextual keyword to introduce the scope of an OCL expression, followed by the name of the type. The keyword *inv*, *pre* and *post* denotes the invariants, pre-conditions and post-conditions of the constraints. After structure translation, design specification need to be mapped by behavior translation, which consists of the following two parts:

- Translations from state schema's predicates into OCL invariants that is denoted by the prefix 'inv'.
- Translations from operation schema's predicates to OCL pre/post-conditions, which is denoted by the prefix 'pre' and 'post'.

Translation of predicates in the state schema is rather straightforward mappings into class invariants. However, the mappings of predicates in the operation schema requires the identifications of pre- and post- conditions. In a post-condition, the expression can refer to two sets of values for a state variable, i.e., the value before and after the operation. If the predicate contains variable followed by the Z prime symbol 'x", it will be a post-condition. In UML/OCL, pre-state of a variable is expressed with suffix @pre, and the post-state of a

variable is expressed as itself. The "@pre" postfix is allowed only in OCL expression that are part of a post-condition.

Operation schemas may have input and output variables, input parameters are denoted with the postfix symbol of '?'; and output parameters are denoted with the postfix symbol of '!' in Z. These input and output variables can be translated to input parameters and return type of the operation schema (method) in UML/OCL.

Finally, in order to translate the entire predicate into its OCL counterpart, we need to provide the mappings for various Z operators.

a) Logic and relational operators: suc as 'Not (¬),And (⋀),Or (⋁),Implies (⇒) and Equivalence (⇔)' can be translated into 'not, and, or, implies and two direction implies in UML/OCL. When we translate A ⇔ B, we should make sure A implies B and B implies A. Relational operators such as '$\leq, \geq, =, \neq$, and if then else' can be directly mapped to '<=,>=, <>, and if-then-else-endif' in UML/OCL.

b) Set operators: could be mapped into operations in the UML/OCL set type, e.g., 's ∈ S', can be represented as S→includes(s) in UML/OCL. And 's∉S' means S→excludes(s). Set Union (∪), Set Intersection (∩), Set Difference (\) maps into →union(), →intersection(), and →reject() separately in UML/OCL. For example, 'products\{p}=products'\{p1}' could be mapped into 'self.products@pre→reject(x|x.code=p.code) =self.products→reject(x|x.code=p1.code)'.

c) Quantifiers: Universal Quantifier (∀) can be translated into the →forAll(...) operation in OCL; and the Existential Quantifier (∃) can be translated into the →exist(...) operation in OCL. For example, '∀p1, p2: products •p1 ≠p2 ⇔ p1.code≠ p2.code' can be translated into UML/OCL as "self.products→forAll(p1,p2|(p1<>p2 implies p1.code<>p2.code) and (p1.code<>p2.code implies p1<>p2))". Because this predicate is in the state schema OLShoppingSys, so it should be translated into invariants in OCL with context of the class OLShoppingSys.

Let us exam another more complete example in Z shown in Figure 2.

$$
\begin{aligned}
&\exists p1 : products' \\
&\quad \bullet\ products \setminus \{p1\} = products \setminus \{p?\} \\
&\quad \land p1.code = p?.code \\
&\quad \land p1.description = p?.description \\
&\quad \land p1.price = p?.price \\
&\quad \land (\textbf{if}\ p?.quantity > 1 \\
&\qquad \textbf{then}\ (p1.quantity = p?.quantity - 1 \land p1.stock = InStock) \\
&\qquad \textbf{else}\ (p1.quantity = 0 \land p1.stock = OutofStock))
\end{aligned}
$$

Fig. 2. Example for quantifies

Firstly, '∃p1:products' maps to 'products→exist(p1|...)', all the other lines after the first line of predicate is a set of boolean expression for p1, which are translated into 'exist()'. The predicate 'products\{p1}=products\{p?}' is translated into'products→exist(p1|self.products@pre→reject(x|x.code=p.code)=self.products→reject(x|x.code=p1.code))', as shown in Figure 3.

```
post AddItempost3:self.products->exists(p1| p1.code=p.code
    and self.products@pre->reject(x|x.code=p.code)
        =self.products->reject(x|x.code=p1.code)
    and p1.description=p.description
    and p1.price=p.price
    and (if  p.quantity >1
         then
         (p1.quantity=p.quantity-1 and p1.stock=StockStatus::InStock)
         else
         (p1.quantity=0 and p1.stock=StockStatus::OutofStock)
         endif))
```

Fig. 3. Translation result for quantifies example

There are two things to be kept in mind, i.e., UML/OCL has a 'endif' after one 'if-else-then' expression finished and enumerations literals can not be invoked directly. It will be write as "Enumerations::literal".

d) Relation operators: Domain and range: dom R is a limitation for domain(A) so in UML/OCL, just limits to A class; ranR is a limitation for range(B) so in UML/OCL just limits B class. Domain and range restriction: R is a relation for A↔B and S ⊆ A and T ⊆ B; then S◁R is the set {(a,b):R|a∈S} R▷T is the set {(a,b):R|b∈T}. In UML/OCL, domain and range restrictions can be translated into invariant constraints imposed onto the association class. Relational composition: R : A ↔ B and S : B ↔ C R ; S ={(a,c):A×C|∃b:B•a R b ∧ b S c}, according to the translation, A, B and C should be three classes and R ; S just means creating a new association class from A to C.

e) Function overriding: f,g:A→B;then f ⊕g is the function (dom g ⩤f)⋃g. As the semantics show that restrictions on the association classes f and g have to be combined. That is, the class A's value must be dom g→union(dom f), and class B's value must be ran f minus the part that dom g points to then →union(ran g).

f) Sequence operators: such as concatenation "~": <a, b> ~<b, a, c> = <a, b, b, a, c> in Z means <a,b>→append(<b,a,c>) in UML/OCL. In Z, sequence also has two important functions, i.e., 'Head' and 'Tail'. Head is to obtain the first element of a sequence return as an element. Tail is to have the rest of sequence's elements except the head. In UML/OCL, for a sequence Q, 'Q→first()' equals to 'Head' and 'Q→excluding(Q→last())' equivalents to 'Tail'.

Based on the above mapping rules, we can translate a Z formal specification into its corresponding UML/OCL model.

III. Translating OCL into Java Assertions

The next step is to validate the code from UML/OCL model. In order to do so we first generate code from the UML model and then insert the assertions for validation purposes. There are many different tools provide code generation from UML model. In this project, we just explored two candidates: Papyrus and OCLinEcore. They both support OCL and code generation. Papyrus can create UML models and OCL constraints graphically, which makes the UML/OCL model visible and easier to understand. It can also generate Java skeleton

code from models and create an basic type of UML model file that can be reused in other tools such as OCLinEcore.

The reason for adding Java assertions and JUnit test cases to code is that to provide extra checking for at the implementation level. Whatever the implementation is, the assertions and test cases can always verify the code correctness. This is a novel contribution that is missing from other code generation approaches.

1) Java Assertion: After generating Java skeleton code using Papyrus, we translate OCL constraints to assertions to ensure program's correctness. A Java assert statement has two expressions: assert <boolean expression> and assert <boolean expression>:<error information>, if boolean expression is true then program can be continue else program will throw java.lang.AssertionError.

All of OCL constraints are boolean expressions, however, some of them can not be represented in Java assertions directly. For example, for a collection such as 'Set' in OCL, code generation tools usually translate it into a list in Java code. Thus methods in set can be expressed by list's operations, then use assertions to check this expressions (with keyword "assert"). The following provides translation guidelines from OCL to Java assertions.

a) Normal boolean expressions: Normal boolean expressions that consist of relational/logical operators and operations on collection types can be mapped into their Java assertions counterparts.

- operators such as: <=,>=, <>,+ and -, are directly supported by Java.
- "=", "not": In OCL constraints "=" translates to "==" in Java and "not" translates to "!" in Java.
- →size(): is to calculate the number of elements in a collection. We use Java List to represent a collection type in OCL. List has a method "size()"to returns the number of elements in the list.
- →includes(object o): checks whether an object is in the collection or not. The "contains(Object o)" in List returns true if this list contains the specified element.
- →excludes(object o): based on the previous rule, "! contains(Object o)" means if the list does not contain the specified element it returns true, which is same with "→excludes(object o)".
- →includesAll(c2:Collection): is to check whether this collection contains all the elements in another collection c2, which is similar to "containsAll(Collection c2)"in Java.
- →excludesAll(c2:Collection): translates to "!containsAll(Collection c2)" in Java.
- →isEmpty(): presents as "isEmpty()" in Java, which will returns true if this list contains no elements.
- →union(c2:Collection): means the result will be a union of the referred collection with c2. In Java, "addAll(Collection c2)" has same meaning as the union.
- →intersection(c2:Collection): is to find out a set of all elements that are in both referred collection and c2. The "retainAll(Collection c2)" does the same in Java.
- - (c2:Collection): equals to the "removeAll(Collection c2)", which takes away the elements in the referred collection, which are in c2.

b) Loops: Apart from normal boolean expressions, there are quantifier related operators which may not have direct mappings into Java assertions expressions, such as:

- →exists(expr:OclExpression)
- →forAll(expr:OclExpression)
- →any(expr:OclExpression)
- →reject(expr:OclExpression)

These constructs above are working on collections that require to examine each individual object against the OCL expression. That is, the operations require to check every object in collection, so a loop structure would be a natural mapping for them. The actual OCL expressions are embedded inside the loop for checking individual values one by one.

2) JUnit Test: OCL not only can be translated to java assertion in java code, but also can generate Junit test case. Junit test can test every method and class separately. It can also simulate any situation for the system. The built-in assertion mechanism of JUnit is provided by the class org.junit.Assert. It provides more static methods than java assertion. The most important method in JUnit test is the "Assert.assertThat([reason,]T actual, Matcher<super T> matcher)", "reason" is a output when this assert fails; "actual" is the real result at assertion; matcher is an matcher for assertion, its logic determines the actual objects whether satisfies assertion or not. Let us see a simple example for creating Junit test case. For the invariant of class OLShoppingSys, we assume a situation after load all the products, every product's code should be unique, the test case is below at Figure 4.

```
@Test
public void testOLShoppingSys() {
    if(products.size()!=1){
        for(int i=0;i<products.size()-1;i++){
            for(int j=i+1;j<products.size();j++){
                Assert.assertThat(products.get(i).getCode(),
                        not(products.get(j).getCode()));
                }
            }
        }
    Assert.assertTrue(products.containsAll(cart.getItems()));
    }
}
```

Fig. 4. Invariants test case

The two invariants are 'self.products→forAll(p1,p2|(p1<>p2 implies p1.code<>p2.code)' and '(p1.code<>p2.code implies p1<>p2))'. In this example, invariants want to check any two variable in products. We create two for loops to compare all the vaules in products' list. For every two values of products list, their code should not be the same. The "Assert.assertThat(products.get(i).getCode(), not(products.get(j).getCode()))" tests if two object references not point to the same object. "assertTrue" has same meaning with keyword "assert", so this invariant can easily be translated from Java assertions in code to test cases.

In summary, by mapping OCL annotations on the UML model into corresponding Java assertions during the code generation process, we are able to provide extra quality assurance/checking on the derived program, as well as for the guided unit testing.

IV. Verification, Simulation and Checking

After the translations, we have two forms of the design that was derived from the Z formal model. In addition to the verification on the Z formal specification itself, e.g., theorem proving with Z-EVES or model checking with ProB, we can further simulate and validate the transformed UML/OCL model with various tools. For the implementation level checking, we can perform validation and debugging with assertions. It is worth noting that the theorem proving and model checking at the Z specification level are complimentary to the simulation and validation at the UML/OCL level. In addition, the assertions at the implementation level further guarantee the code correctness.

A. Design level validation with OCL

Now we have a UML model with OCL constraints, we can simulate it and dynamic validate the design by creating object instants and checking the static properties, invariants and pre/post-conditions. The UML Specification Environment (USE) [9] is a simulation and validation tool for OCL specifications. After USE load the OCL model, it can display a view of the class diagram of the system. USE provides simulation facilities by creating objects and association instances of the model.

After creating the system states, USE can check the OCL invariants by class invariant view. In the example below, when we set 'products1' and 'product2' codes both to '1001', the result of invariants checking is false due to the unique code restriction. After we change 'product2' code to '1002', the checking result is true, as shown in Figure 5.

Fig. 5. Invariants checking result in USE

Pre/post-conditions can be validated in USE. It simulates the process of invoking the operation for validating the pre/post-conditions. Here is a example for the success and failure of pre-conditions of the 'DeleteItem' operation. As shown in Figure 6, if user want to delete an item from the shopping cart, the item must be in the cart. For instant, after adding 'product1' to cart, the user can not delete 'product2' from cart, the pre-condition will be false, due to the 'product2' has not been added to cart. However, the user can delete 'product1' from cart. USE also has a functionality to record a sequence of operations by the sequence diagram view. This function serves as a simulation traces for the real user operations.

```
use> !openter ol deleteItem(product2)
precondition `DeleteItempre1' is false
Error: precondition false in operation call `OLShoppingSys::deleteItem(self:@ol,
 p:@product2)'.
use> !openter ol deleteItem(product1)
precondition `DeleteItempre1' is true
```

Fig. 6. Success and failure example for deleteItem

B. Code level checking with assertions

The reason that we translate OCL constraints to Java assertions and JUnit test is to validate and check the implementations of classes and operations. Java assertions and JUnit test cases have same meaning with Z predicate and OCL constraints, and they can both check invariants and pre/post-conditions of a design model.

1) Java Assertion: Assertion is a normal debug style, and lots of languages support this function like C, C++, Eiffel and so on. In syntax, Java provides a keyword 'assert' to insert assertion. Java uses '-enableassertions/disableassertions' to open/close assertion function. When a programming closes assertion function, assertion expression will lose their effects. If assertion function is open, then java will calculate the expression value. If the value is false, then this expression will throw a AssertionError.

Here is an example for using Java assertion in the online shopping system, which checks assertions in the addItem operation. The error denotes that input 'p' may not in the products list or currently out of stock, which was guaranteed in the assertion definition. If the item code does not meet this requirement, the system will not invoke the addItem method. For instant, we add a product (code is 1018) to cart, which is out of stock, the result will throw an AssertionError, as is shown at Figure 7.

```
================================================================
----------------------------------------------------------------
1. Add an item to the shopping cart.
2. Delete an item from the shopping cart.
3. Check out the shopping cart.
4. Exit without buying.
----------------------------------------------------------------
Please enter your choice: 1
----------------------------------------------------------------
Enter item code to buy: 1001
One more- Fresh toast bread white (700g) is added to the shopping cart
the items quantity in cart is:1
================================================================
----------------------------------------------------------------
1. Add an item to the shopping cart.
2. Delete an item from the shopping cart.
3. Check out the shopping cart.
4. Exit without buying.
----------------------------------------------------------------
Please enter your choice: 1
----------------------------------------------------------------
Enter item code to buy: 1018
OutofStockException in thread "main"
java.lang.AssertionError: this product is out of stock
        at OLShoppingSys.addItem(OLShoppingSys.java:309)
        at OLShoppingSys.main(OLShoppingSys.java:128)
```

Fig. 7. AddItem operation assertions

2) JUnit Test: JUnit testing can be performed independently of the the actual implementation. Eclipse supports JUnit test cases generated from Java code by right-click choose

new JUnit test case. Users can also choose which classes and methods they want to test in the set. JUnit test case can be run by choosing to run as 'JUnit Test' in Eclipse. Several test be created cases for checking online shopping system's classes and methods, as is shown at Figure 8.

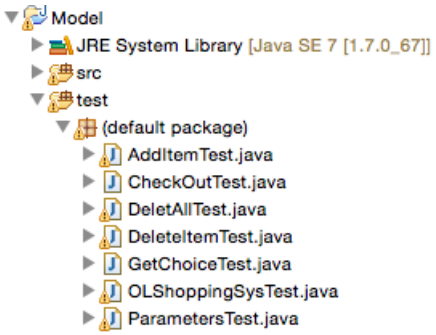


Fig. 8. Test cases for online shopping system

Let us exam an example of simulating a situation of adding items to the shopping cart, as is shown in Figure 9. This test case is not just testing the addItem method, it also tests the getChoice method. It prints out the choices and wait for the user input. When the user inputs '1', the system will know that the user would like to add an item to cart, it will ask for input of the product code. If the item code exists in the product list, the result will be successful. In this test case, we can check two parts: one for input product code, another is for the input choice. If the product code is not valid or its stock status is not 'InStock', the result will be failure. If the input choice is not '1', which means the user does not want to add an item, the result will also be failure in this particular test case.

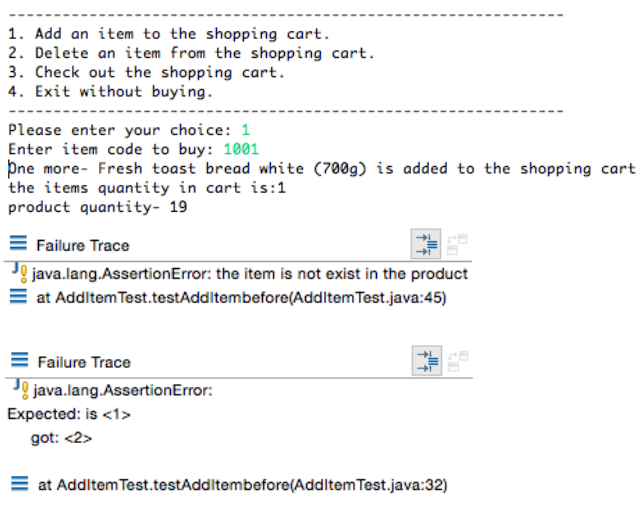


Fig. 9. JUnit testing for the AddItem method

The above case study of the online shopping system aims to validate the feasibility of our approach. The associated tools, such as Z/EVE, ProB, USE, Java assertions and JUnit test cases also provide extra evaluations to the results of each steps in our approach. We use the runnable case to demonstrate all translations as well as verification results. Therefore, the case study provides the evaluation for our approach, which is feasible and effective.

V. Conclusions

In this paper, we explore the possibilities of automatically derive implementation from formal design models. We translate Z specifications into UML/OCL and then map them into Java code. We use Java assertions to express OCL constraints at the code level. Java assertions can be used to generate JUnit test cases that are independent from the implementation and can be run repeatedly. Java assertions act as extra checking facilities to make sure the code correctness. At the each level of translation, user can verify, simulate, validate and debugging the model/code to ensure the quality of the final product.

Due to the limitation of time, some improvements for our project can be made in the future. We defined the translation guidelines from Z to UML/OCL, OCL to Java assertions. However, the translation is still manual based. It would be beneficial to develop a tool to provide the automated transformations. Since OCLinEcore and JUnit have Eclipse plug-in tools, and Z/EVE has an Eclipse plug-in version in CZT, there is a possibility to create an Eclipse plug-in tool that integrates the associated tools into one coherent interface. Thus, the entire design modelling, transformation, code generation and testing could all be achieved in Eclipse.

References

[1] Monin, J. F. (2012). Understanding formal methods. Springer Science & Business Media.

[2] Vogel-Heuser, B., Witsch, D., & Katzke, U. (2005, June). Automatic code generation from a UML model to IEC 61131-3 and system configuration tools. In Control and Automation, 2005. ICCA'05. International Conference on (Vol. 2, pp. 1034-1039). IEEE.

[3] Budinsky, F. J., Finnie, M. A., Vlissides, J. M., & Yu, P. S. (1996). Automatic code generation from design patterns. IBM systems Journal, 35(2), 151-171.

[4] Gray, J., Tolvanen, J. P., Kelly, S., Gokhale, A., Neema, S., & Sprinkle, J. (2007). Domain-specific modeling. Handbook of Dynamic System Modeling, 7-1.

[5] Hinchey, M. G., Rash, J. L., & Rouff, C. A. (2005). Requirements to design to code: Towards a fully formal approach to automatic code generation. NASA tech. monograph TM-2005-212774, NASA Goddard Space Flight Center.

[6] Massol, V., & Husted, T. (2003). Junit in action. Manning Publications Co.

[7] Hui Liang, Jin Song Dong, Jing Sun and W. Eric Wong. "Software Monitoring through Formal Specification Animation". *Innovations in Systems and Software Engineering: A NASA Journal*, Volume 5, Issue 4, pages 231-241, Springer, December 2009.

[8] Cabot, J., & Gogolla, M. (2012). Object constraint language (OCL): a definitive guide. In Formal Methods for Model-Driven Engineering (pp. 58-90). Springer Berlin Heidelberg.

[9] Gogolla, M., Büttner, F., & Richters, M. (2007). USE: A UML-based specification environment for validating UML and OCL. Science of Computer Programming, 69(1), 27-34.

[10] Sun, J., Dong, J. S., Liu, J., & Wang, H. (2001, April). Object-Z web environment and projections to UML. In Proceedings of the 10th international conference on World Wide Web (pp. 725-734). ACM.

[11] Long, Q., Liu, Z., Li, X., & Jifeng, H. (2005, April). Consistent code generation from UML models. In Software Engineering Conference, 2005. Proceedings. 2005 Australian (pp. 23-30). IEEE.

[12] H. Zhu, J. Sun, J. S. Dong, and S.-W. Lin, "From Verified Model to Executable Program: the PAT Approach," *Innovations in Systems and Software Engineering*, pp. 1–26, 2015.

Self-adaptive Systems Driven by Runtime Models

A Systematic Literature Review of Approaches

Dr. Marcello Thiry
Technological Science Center of Earth and Sea (CTTMar)
University of Vale do Itajaí (UNIVALI)
Florianópolis, Brazil
marcello.thiry@gmail.com

Roger Anderson Schmidt
Technological Science Center of Earth and Sea (CTTMar)
University of Vale do Itajaí (UNIVALI)
Florianópolis, Brazil
rasflp@gmail.com

***Abstract*— Model-Driven Software Engineering (MDSE) represents a promising research area with a variety of challenging issues open for discussion. Expanding the limits of the MDSE paradigm, runtime models keep abstract representations of the running system in order to trigger on-the-fly software reconfigurations. One of the most popular applications of runtime models are self-adaptive systems, since abstractions can be fine-tuned not only in the development phases, but also in runtime. As this kind of system needs to modify its behavior during execution, this can be achieved by means of high-level model interventions. The objective of this article is to present relevant approaches of self-adaptive systems driven by runtime models. This article can help practitioners to get an overall picture of current approaches, in terms of methods, techniques and tools. Researchers can also be inspired to create new or to extend current approaches, facing the challenges identified here. To that end, we conducted a rigorous Systematic Literature Review based on the guidelines proposed by Kitchenham. This paper provides answers for four research questions, based on 16 selected articles. In the conclusion, we present some considerations and challenges based on the results obtained from this review.**

Keywords-Self-Adaptive System; Runtime Model; Model-Driven Software Engineering; Systematic Literature Review

I. INTRODUCTION

Modern software execution environments have become increasingly decentralized, heterogeneous, uncertain and changing. Fully adequate for these scenarios, mobile sensor-based devices have been providing significant computational power in various domains. From their assorted sensors, a huge amount of data can be collected resulting in a rich environmental context. Reacting to changes in this context, mobile applications for remote environments, such as daily life objects from the Internet of Things (IoT) [1] or small devices connected to Wireless Sensor Networks [2] should dynamically adapt themselves to a new external configuration. Thanks to this, Context-Aware and Mobile (CAM) applications [3] that explore Pervasive Computing techniques [4] have been receiving special interest in the recent years.

This background has been leading the software engineering community to propose innovative ways for building, running and managing systems and services [5]. Besides that, the boundaries between design and runtime have to be changed, as designers can not anticipate all possible circumstances that might appear during the execution of an application [1].

In order to meet those demanding expectations, dynamic adaptive systems represent a turning point for responding to changes, as software becomes capable of reconfiguring itself, without the need of being rebuilt. In addition, systems should be able to adapt its structure and/or behavior in response to changes in the execution context and varying user needs. For this purpose, dynamic reconfiguration should be applied at runtime whenever it is needed [5] [6] [7].

However, a particularly important problem arises from the complexity to manage self-adaptive systems. A promising approach to deal with this complexity is to develop adaptation mechanisms that leverage software models, referred to as models@run.time, or runtime models [8]. Conceptually, a runtime model is defined as an abstraction of a running system that is being manipulated during its execution for a specific purpose. Runtime models are also a causally connected self-representation of the system that emphasizes the structure, behavior or goals from a problem space perspective [9].

Therefore, the combination of runtime models and self-adaptive systems opens the possibility of using abstractions at different levels to change the behavior of running systems. The "model-driven" approaches based on runtime models raise the importance of modeling activities. As identified by [9] [10] and several others, both topics present important problems to be addressed, which increases the relevancy of research in this field.

In this paper, a literary research was undertaken on the proposals of approaches for development, execution, change monitoring and reconfiguration of self-adaptive systems through runtime models. For the purpose of clarification, the understanding of an "approach" assumed by this paper is characterized in Sect. II.C. In order to identify, evaluate and interpret all the available papers relevant to our research questions, we choose the Systematic Literature Review (SLR) research method.

This article is structured as follows: Sect. II provides some background about the research topics. Sect. III presents some summarized information about the research method definition that guided this review. In Sect. IV, the outcomes from the research process execution are exposed. Sect. V shows the data extraction and synthesis. Then, the next section presents analysis and discussion based on the research questions. Finally, we conclude this review in Sect. VII.

DOI reference number: 10.18293/SEKE2017-168

II. BACKGROUND

A. Runtime Models inside the MDSE Context

In the literature, there is no consensus about the concept of MDSE. MDSE can be defined as a methodology for applying the advantages of modeling to software engineering activities, which comprises the following aspects: concepts, notations, process and rules, and tools [11]. MDSE is also defined as a family of development processes that focuses on the model as primary development artifact [12].

Traditionally, the MDSE area has primarily focused on using models at design, implementation and deployment phases of the software development life cycle. However, as systems become more adaptable, reconfigurable and self-manageable, runtime models are needed to tackle the complexity of dynamic adaptations by keeping an abstract model of the running system. It pushes the idea of reflection one-step further by synchronizing the abstract model with the actual system, so a change performed on the model is automatically accommodated by the system [1].

B. Self-Adaptive Systems (SAS)

A dynamically adaptive system should be able to adapt its structure and/or behavior in response to changes in the execution context and varying user needs. For this purpose, dynamic re-configuration should be applied at runtime whenever it is needed, in an anticipated or unanticipated form. This determines if the reconfiguration is caused by expected changes in requirements, so it can be considered and planned before being needed, otherwise it is impossible to predict [7].

In addition, reconfiguration actions can be either architectural or behavioral. Architectural reconfigurations consist of modifying the system structure such as add, remove, start, stop, replace and migrate components/connections. On the other hand, behavioral reconfigurations are limited to modify properties of components or connections [7] .

C. Characterization of an "Approach" for this SLR

This paper considers an approach the documented integration of the following components:

- *Process:* an overall workflow definition
- *Abstractions:* meta-meta-models, meta-models, models
- *Transformations:* models-to-text, text-to-models, final or intermediate code generation
- *Self-adaptive Infrastructure:* framework, middleware, or model-based approach [13] that provides low-level services for SAS development and execution
- *Tools*: developed or integrated supporting tools

III. DEFINITION OF THE SYSTEMATIC LITERATURE REVIEW

Our research methodology is inspired by Kitchenham et al. guidelines [14] and procedures [15], which are specifically proposed for Systematic Literature Reviews (SLR) in software engineering. To that end, three main phases are defined: 1. Planning the Review (identify the need for a review and develop a review protocol – Sect. III); 2. Conducting the Review (identify the research, select primary studies, assess the study quality, extract, monitor and synthesize the data – Sects. IV and V); and 3. Reporting the Review (present the results of the review and its dissemination to the interested parties – Sects. VI and VII). We started from the identification of the need for a review, as it follows.

A. Related Work

When starting the planning phase of our research methodology, we checked the need for a systematic review. Therefore, a search for SLRs on the main topics was conducted online through selected databases, from 2012 to 2017 (up to Jan 30). The basic terms in the search string for SLRs were "systematic literature review", "self-adaptive", "model-driven" and several other variations. As a result, we found 55 reviews. After grouping and removing the duplicates, 31 distinct SLRs remained. Then, the SLRs were filtered by metadata evaluation, a stage in which 28 were dismissed for being out of context (8) or too specific in topics such as, security (4), processes (3), DSLs (2), formal verifications (2), and another 9 different topics.

Finally, 3 SLRs remained for full text evaluation. Firstly, the review conducted by Svetits and Zdun [9] presented a comprehensive research on models at runtime literature, classifying the articles in terms of: objectives, architectures, techniques, and kinds of models. Secondly, the SLR conducted by Giachetti et al. [16] focused on interoperability in MDD processes, by evaluating five related features from selected approaches. Finally, Becker et al. [17] focused on model-driven performance engineering, classifying approaches into adaptation, architecture, performance analysis, and applicability criteria.

In contrast to the cited works, this review is exclusively focused on self-adaptive systems driven by runtime models, with the adoption of a well-defined criteria of what is considered an approach. Besides that, its covers articles published until 2017.

B. Research Questions

As suggested by [14] and [16], the PICO(C) criteria [18] was applied in order to frame and structure our research questions. The values of each criterion are the following:

- *Population:* Domain experts, software architects, and systems analysts. In short, model designers.
- *Intervention:* Approaches for dynamic reconfigurations of self-adaptive systems through runtime models.
- *Comparison:* evaluation of different approaches applying defined criteria.
- *Outcomes:* processes, methods and techniques, model types and transformations, reconfiguration strategies, context-awareness and consistency checking, etc.
- *Context:* development and execution phases of self-adaptive systems.

After taking into consideration the five viewpoints presented in the PICO(C) model, it became easier to identify the research questions that follow:

RQ1. What is the central point of each proposal in order to support dynamic reconfigurations of self-adaptive

systems through runtime models? What are the most predominant methods or techniques applied?

RQ2. What levels of abstraction are provided for the model designer? Do the studies suggest an assignment of modeling tasks to different roles according to their skills (e.g. domain expert, software engineer)?

RQ3. Do the studies present an overview of the suggested process? How are they composed in terms of metamodeling, model languages and transformations (code generation included) and related tools?

RQ4. What are the strategies for model changes monitoring and adaptive system dynamic reconfigurations? Does the adaptation engine implemented by or integrated with the studies support anticipated and unanticipated context changes?

C. Literature Selection Criteria

During the research protocol definition, we specified a reliable study selecting criteria in order to ensure that all primary studies provide direct evidence about the research question [14].

1) Inclusion criteria

- Articles published from January 1, 2012 to January 20, 2017 (around 5 years range)
- Articles found in selected electronic databases
- Articles published in peer-reviewed journals, conferences and workshops

2) Exclusion criteria

- Studies not reported in English
- Publications without abstracts
- Books, web sites, technical reports, and master thesis
- Publications in which the research topics are not clearly established and documented, or that explore the term "model" outside the context of software development

D. Data sources

The search strategy for this review included four electronic databases (TABLE I), selected in terms of relevancy in the research areas of Computer Science and Engineering.

TABLE I. Results from search on databases

Database	Total amount of records [a]	SLRs (the need of a review)	Primary Studies (initial search)
SCOPUS (Elsevier)	Over 60 million	21	111
Association for Computing Machinery (ACM) Guide to Computing Literature	2,625,656	17	71
IEEE Xplore Digital Library	4,145,171	10	33
ScienceDirect (Elsevier)	Over 14 million	7	9
Total		**55**	**224**

a. Data collected on January 30, 2017

E. Search String Composition

After the database selection, we started to compose the search string by defining its components and keywords. Besides, we had to explore thoroughly every engine mechanism in order to produce a consistent query for each database [19]. During this process, we were aware of the variations of the research concepts, as shown in TABLE II. After many testing iterations, our search string was defined by aggregating the key topics for this review.

TABLE II. Search keywords

Concept	Keyword and synonymous
P = Self-Adaptive Systems	"self-adaptive", "self adaptive", "adaptive system", "adaptive software"
Q = Model-Driven Software Engineering	"model-driven", "model driven", "mde", "mdse", "mdd"
S = Runtime Models	"models at runtime", "models@runtime", "models@run.time", "models for runtime", "runtime models"
Result	**(P ∨ Q ∨ S)**

IV. Execution of the SLR

A. Literature Search Process

The literature search process that guided our review is presented in Figure 1, which shows the outcomes from every activity of the process, highlighting the articles selected and dismissed due to related reasons. Initially, we submitted the final versions of them to each selected electronic database (1). Then the metadata of all returned articles (title, abstract and keywords) was exported to BibTeX format, which is accepted by the Mendeley import tool. Next, we proceeded with a careful metadata checking of every single article against the literature selection criteria (2). As a result, 150 articles were selected. After grouping and duplicates removal (3), 134 articles remained.

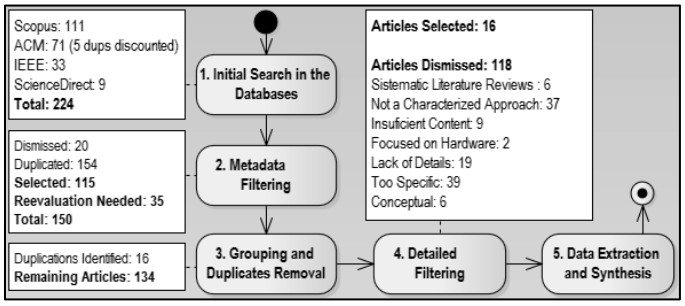


Figure 1. Literature Search Process

The final selection of articles was performed after full text reading and evaluation (4). In this stage, we especially focused on the compliance with the approach characterized in Sect. II.C. As a result, the 16 articles presented in TABLE III were selected for data extraction and synthesis (5).

TABLE III. Selected Studies

ID	Reference
S1	S. Loukil, S. Kallel, and M. Jmaiel, "An approach based on runtime models for developing dynamically adaptive systems," *Futur. Gener. Comput. Syst.*, vol. 68, pp. 365–375, 2017.

ID	Reference
S2	F. Moyano, C. Fernandez-Gago, and J. Lopez, "A Model-driven Approach for Engineering Trust and Reputation into Software Services," *J. Netw. Comput. Appl.*, vol. 69, no. C, pp. 134–151, 2016.
S3	B. Djoudi, C. Bouanaka, and N. Zeghib, "A formal framework for context-aware systems specification and verification," *J. Syst. Softw.*, vol. 122, pp. 445–462, 2016.
S4	J. M. T. Portocarrero, F. C. Delicato, P. F. Pires, T. C. Rodrigues, and T. V. Batista, "SAMSON: Self-adaptive middleware for wireless sensor networks," in *Proceedings of the ACM Symposium on Applied Computing*, vol. April 04–08, pp. 1315–1322, 2016.
S5	M. Hussein, R. Nouacer, and A. Radermacher, "A Model-Driven Approach for Validating Safe Adaptive Behaviors," in *Proceedings of the 19th Euromicro Conference on Digital System Design*, pp. 75–81, 2016.
S6	J. Yu, Q. Sheng, J. K. Y. Swee, J. Han, C. Liu, and T. H. Noor, "Model-driven development of adaptive web service processes with aspects and rules," *J. Comput. Syst. Sci.*, vol. 81, no. 3, pp. 533–552, 2015.
S7	P. A. de S. Duarte, F. M. Barreto, F. A. de A. Gomes, W. V. de Carvalho, and F. A. M. Trinta, "CRITiCAL: A Configuration Tool for Context Aware and mobiLe Applications," in *Proceedings of the 2015 IEEE 39th Annual Computer Software and Applications Conference* – Volume 02, pp. 159–168, 2015.
S8	J. Bocanegra, J. Pavlich-Mariscal, and A. Carillo-Ramos, "MiDAS: A model-driven approach for adaptive software," in *Proceedings of the WEBIST 2015 – 11th International Conference on Web Information Systems and Technologies*, pp. 281–286, 2015.
S9	D. B. Abeywickrama, N. Hoch, and F. Zambonelli, "An integrated Eclipse plug-in for engineering and implementing self-adaptive systems," in *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 3–8, 2014.
S10	M. Hussein, J. Han, J. Yu, and A. Colman, "Enabling Runtime Evolution of Context-Aware Adaptive Services," in *Proceedings of the 2013 IEEE International Conference on Services Computing*, pp. 248–255, 2013.
S11	M. Luckey and G. Engels, "High-quality specification of self-adaptive software systems," in *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pp. 143–152, 2013.
S12	N. Ferry, F. Chauvel, A. Rossini, B. Morin, and A. Solberg, "Managing multi-cloud systems with CloudMF," in *ACM International Conference Proceeding Series*, pp. 38–45, 2013.
S13	C. Ghezzi, L. S. Pinto, P. Spoletini, and G. Tamburrelli, "Managing non-functional uncertainty via model-driven adaptivity," in *Proceedings of International Conference on Software Engineering*, pp. 33–42, 2013.
S14	J. Floch, C. Frà, R. Fricke, K. Geihs, M. Wagner, J. Lorenzo, E. Soladana, S. Mehlhase, N. Paspallis, H. Rahnama, P. A. Ruiz, and U. Scholz, "Playing MUSIC - Building context-aware and self-adaptive mobile applications," *Softw. - Pract. Exp.*, vol. 43, no. 3, pp. 359–388, 2013.
S15	S. Hallsteinsen, K. Geihs, N. Paspallis, F. Eliassen, G. Horn, J. Lorenzo, A. Mamelli, and G. A. Papadopoulos, "A development framework and methodology for self-adapting applications in ubiquitous computing environments," *J. Syst. Softw.*, vol. 85, no. 12, pp. 2840–2859, 2012.
S16	M. Amoui, M. Derakhshanmanesh, J. Ebert, and L. Tahvildari, "Achieving dynamic adaptation via management and interpretation of runtime models," *J. Syst. Softw.*, vol. 85, no. 12, pp. 2720–2737, 2012.

V. Data Extraction and Synthesis

In consonance with the research questions presented in Sect. III.B, relevant information was extracted from the selected studies. The overall answers to the first question (RQ1) are presented in Table IV. In relation to the third question (RQ3), Table V and Table VI present data extracted.

Table IV. Overview of the selected studies (RQ1)

ID	Central Point	Main Methods or Techniques
S1	Middleware responsible for monitoring the system and performing architectural reconfiguration by Aspect Oriented Software Development. Concurrency between reconfigurations is supported.	Architecture Description Language (ADL), AOSD
S2	Framework for trust and reputation that allows developers to implement different types of security models in a high level of abstraction thanks to the usage of metamodeling techniques.	Kevoree Distributed Dynamic Component Model Integration
S3	Framework for specification and verification of context-aware systems. A DSL is provided to allow designers to specify context entities, states and actions. It also provides a tool with several features.	CBSE, Formal Methods, Maude-based DSL
S4	Process to generate an instance of a Reference Architecture from its specification. Model-driven transformations are used to map elements and to generate the source code to be deployed in a WSN platform.	Reference Architecture (RA), Middleware Instantiation, MAPE-K
S5	Approach to facilitate the validation of the adaptive behavior of embedded software in the fully electric vehicles domain. Fault injection and monitoring techniques are applied on a virtual self-adaptive platform.	Architecture Description Language (ADL)
S6	Approach to support the development of dynamically adaptive WS-BPEL based systems. To that end, an aspect-oriented method is developed in order to perform runtime changes.	AOSD, WS-BPEL, Web Ontology Language (OWL)
S7	Approach for developing Context-Aware and Mobile (CAM) applications, by modelling contextual information and rule-based behaviour by using a visual notation.	OSGi, Middleware (LOCCAM), DSL
S8	Framework that provides a new language for requirements (with uncertainty support), a method to derive concrete implementations in specific architectures, besides a mechanism for traceability and sincronization.	DSL, Traceability
S9	An integrated Eclipse plug-in tool to architect, engineer and implement self-adaptive systems through a feedback loop-based approach. Also provides modeling, simulation and code generation features.	FCL-Based, MAPE-K
S10	Approach to enable runtime evolution of context-aware adaptive services in response to unanticipated changes in their environments or functionalities. Differences between running and its evolved model are computed.	SAS Runtime Evolution
S11	Method for software specification by using UML based concern-specific modeling language. It allows separated and explicit specification of self-adaptivity concerns.	SoC, MAPE-K
S12	Framework for modeling dynamically cloud-based adaptive systems by enabling adaptation at runtime. It consists of a tool-supporting DSL and a models@runtime environment.	Cloud Computing, DSL
S13	Framework to support the development and execution of software that tolerates manifestations of uncertainty, in order to satisfy certain non-functional requirements divided into two classes Threshold-based and Max/Min.	Probability Theory, Markov Decision Process (MDP)
S14	Description of typical context and adaptation features relevant for the development of context-aware and self-adaptive mobile applications, based on several demonstrations of the MUSIC adaptation framework.	Context-awareness, Self-adaptation, MUSIC framework
S15	Discussion of the motivation, technical approach, and results of the MUSIC project, which provides a software development framework for self-adaptive applications that operate in ubiquitous and dynamic environments.	OSGi, MUSIC framework, MAPE-K
S16	Approach for realizing fine-grained dynamic adaptation in software systems by managing and interpreting graph-based models of software at runtime. Includes a comprehesive case study presented in detail.	AOSD, MAPE-K, TGraph Approach

Table V. External tools applied (RQ3)

Name	Type	Studies
Ocarina	Distribute Applications Generator	S1
Kevoree	Distributed Reconfigurable Software Dev.	S2
Papyrus	Model-Based Engineering tool	S5
Acceleo	Transformation Tool	S3
UNISIM-VP	Virtual Platform for Simulator Environment	S5
Drools	Business Rules Engine	S6
BPEL	Execution Language Engine	S6
LOCCAM	Self-Adaptive Middleware	S7
JET	Template Engine for Code Generating	S9
ROAD	Apache Axis2 Extension for adapting services	S10
PRISM	Probabilistic Model Checker	S13
MUSIC	Self-Adaptive Framework	S14, S15
MOFScript	Transformation Builder	S14, S15
JGraLab	API for Processing TGraphs	S16

Table VI. Languages and Notations applied (RQ3)

Name	Purpose	Studies
UML	General-purpose Modeling	S5, S9, S13, S16
OWL	Semantic Web	S6, S14, S15
OCL	Object Constraint	S1, S3
Drools	Business Rule	S6, S10
AADL	Architecture Analysis and Design	S1
AO4AADL	Aspect Oriented Extension for AADL	S1
Schematron	Rule-based XML Validation	S1
Kevscript	Kevoree Reflection Layer Scripting	S2
Pi-ADL	Formal Description	S4
EAST-ADL	Architecture Description	S5
BPMN	Business Process Description	S6
UML AL	UML Action Profile	S9
DMM	Graph-transformation	S11
ATN	Automata Theory	S13
MUSIC	UML Profile for Adaptation Modeling	S15
GReQL	Graph Repository Query Language	S16
GReTL	Graph Repository Transformation Language	S16

VI. ANALYSIS AND DISCUSSION

In this section, we analyzed each research question with the purpose of achieving the defined goals.

RQ1. What is the central point of each proposal in order to support dynamic reconfigurations of self-adaptive systems through runtime models? What are the most predominant methods or techniques applied?

A recurring method found in the studies is the MAPE-K reference model, which served as basis for their autonomic feedback loop implementation (S4, S9, S11, S15, S16). It is also important to highlight that the Component-Based Software Engineering (CBSE), in general or specifically related to the OSGi dynamic component model, had significate representation among the studies (S2, S3, S6, S7, S15, S16). This approach has been commonly used to support the Separation of Concerns (SoC) design principle and to provide dynamic architectural reconfigurations in the proposed self-adaptive systems. Towards similar goals, Aspect Oriented Software Development (AOSD) is found in a representative number of studies as well (S1, S6, S16).

RQ2. What levels of abstraction are provided for the models designer? Do the studies suggest an assignment of modeling tasks to different roles according to their skills (e.g. domain expert, software engineer)?

Ten studies present a high-level business model for runtime interventions (S1-S8, S10, S13). In general, the studies propose a clear separation between business logic, context awareness and adaptation concerns, by using models (S1, S3, S5, S6, S7, S10, S15), API (S2), reference architecture (S4) and language (S11). Two studies present a specific model for invariants specification (S1, S3). Less than half of the studies suggest well-defined separated roles for each abstraction level (S1, S4-S8, S11). In relation to the number of architectural layers, studies present two (S1, S2, S5, S7, S13), three (S4, S6, S8, S10) and four (S3, S16).

RQ3. Do the studies present an overview of the suggested process? How are they composed in terms of metamodeling, model languages and transformations (code generation included) and related tools?

Self-adaptive systems driven by runtime models require at least a two-stage process, since there are not only development phases, but also runtime dynamics when the system's reconfiguration is expected to happen. In general, studies present a process in the form of an overall workflow (big picture discussed in detail) for both introduced stages (S1, S3-S7, S10, S11, S13, S16). In relation to the number of model transformations, the methods presented by the studies include; only one (S3, S5, S7, S8), two (S2) or three (S1, S6). Code generation, which represents a special kind of model transformation, is explored by the studies aiming to a platform/programming language, as follow: Java/Java (S1, S2, S5, S7, S9, S13-S16), AspectJ/Java (S1) and Contiki/C (S4). Among UML artifacts, Activity Diagram is the most popular representation applied in the studies as a source for code generation (S9, S13, S16). Another core concept of MDSE is metamodeling, which techniques are widely used by the studies (S1-S5, S7, S10-S12, S16). Some approaches implement Domain Specific Languages from scratch (S3, S6, S7, S8, S11). Besides that, several studies provide in-house developed tools for models, in concern of design (S1, S3, S6, S7, S8, S9, S10, S12, S14 and S15) and transformation (S3, S6, S8, S10, S13, S14, S15). To that end, most of these tools are Eclipse EMF-based. Finally, S1 and S3 present sound techniques for invariants specification.

RQ4. What are the strategies for model changes monitoring and adaptive system dynamic reconfigurations? Does the provided or suggested adaptation engine support anticipated and unanticipated context changes?

The most identified strategy for model change monitoring and adaptive system dynamic reconfigurations is the implementation of a middleware or framework (S1-S3, S5, S8, S12-S16). Formal methods are used in S3 and S13, but the latter generates an automaton from UML Activity Diagrams, while the former defines semantics for a DSL to formalize context-aware systems structure and behavior. In relation to reconfiguration actions, architectural (S1, S2, S3, S4, S7, S16) and behavioral (S3, S13, S6) were recognized. Finally, adaptation engines of S1, S3 and S10 can cope with unanticipated context changes.

A. Threats to Validity

Publication bias represents a serious threat to the validity of the conclusions produced by a SLR, since it is likely that published studies will have more 'positive' results, that is, failures are seldom reported. A common threat during the search process is not finding relevant studies. We minimized this risk by selecting four comprehensive databases. Besides that, we composed a search string with several variations, and we gave special attention to details of each query tool provided by the selected databases.

With respect to the influence of personal researchers' opinions during the selecting process, we defined a very clear literature selecting criteria during the SLR planning phase. Lastly, some information not explicitly described in the articles had to be inferred by the authors. To minimize this threat, we promoted review sessions, when conflicting or unclear interpretations were discussed cooperatively.

VII. Conclusion

This article presents an overview on approaches for development and execution of self-adaptive systems driven by runtime models. As a result of this Systematic Literature Review, 16 relevant articles were considered, after analyzing more than 220 initial studies.

From what we observed, the method of consistency checking between model and system at runtime is not clearly discussed in the majority of the analyzed articles. Regarding this problem, few studies demonstrated a rollback behavior when the change invalidated a constraint or invariant. Consistency checking is noted by some authors as an important challenge in this field.

Another challenge relies on using the adequate abstract models to define and perform dynamic reconfiguration. UML and its extension mechanisms have been widely used in this direction. Ontologies and Business Process Models have been explored also. In order to increase the model expressiveness, we considered the usage of different languages and notations extremely positive.

Finally, there is a limited range of options available when it comes to environments for distributed reconfigurable software development and execution. Furthermore, the existent ones need to evolve to support traceability, unanticipated changes (uncertainty levels), models reusability and low-level services for self-adaptive systems implementation and monitoring.

In reference to future work, we plan to consider more articles through the "snowballing" practice from references of the selected papers. Besides this, aligning the research protocol towards microservices-based approaches as a provider for architectural reconfigurations, is planned.

References

[1] F. Moyano, C. Fernandez-Gago, and J. Lopez, "A Model-driven Approach for Engineering Trust and Reputation into Software Services," *J. Netw. Comput. Appl.*, vol. 69, no. C, pp. 134–151, 2016.

[2] J. M. T. Portocarrero, F. C. Delicato, P. F. Pires, T. C. Rodrigues, and T. V. Batista, "SAMSON: Self-adaptive middleware for wireless sensor networks," in *Proceedings of the ACM Symposium on Applied Computing*, vol. 04–08, pp. 1315–1322, 2016.

[3] P. A. de S. Duarte, F. M. Barreto, F. A. de A. Gomes, W. V. de Carvalho, and F. A. M. Trinta, "CRITiCAL: A Configuration Tool for Context Aware and mobiLe Applications," in *Proceedings of the 2015 IEEE 39th Annual Computer Software and Applications Conference - Volume 02*, pp. 159–168, 2015.

[4] B. Djoudi, C. Bouanaka, and N. Zeghib, "A formal framework for context-aware systems specification and verification," *J. Syst. Softw.*, vol. 122, pp. 445–462, 2016.

[5] B. H. C. Cheng, R. De Lemos, H. Giese, P. Inverardi, and J. Magee, *Software Engineering for Self-Adaptive Systems*. 2009.

[6] S. Wätzoldt and H. Giese, "Classifying distributed self-∗ systems based on runtime models and their coupling," in *CEUR Workshop Proceedings*, 2014, vol. 1270, pp. 11–20.

[7] S. Loukil, S. Kallel, and M. Jmaiel, "An approach based on runtime models for developing dynamically adaptive systems," *Futur. Gener. Comput. Syst.*, vol. 68, pp. 365–375, 2017.

[8] G. Blair, N. Bencomo, and R. B. France, "MODELS@RUN.TIME Introduction," *Computer (Long. Beach. Calif).*, vol. 42, no. 10, pp. 22–27, 2009.

[9] M. Szvetits and U. Zdun, "Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime," *Softw. Syst. Model.*, pp. 1–39, 2013.

[10] M. Goulão, V. Amaral, and M. Mernik, "Quality in model-driven engineering: a tertiary study," *Softw. Qual. J.*, vol. 24, no. 3, pp. 601–633, 2016.

[11] M. Brambilla, J. Cabot, and M. Wimmer, *Model-driven software engineering in practice*. Morgan & Claypool Publishers, 2012.

[12] N. Macedo, T. Jorge, and A. Cunha, "A Feature-based Classification of Model Repair Approaches," *IEEE Trans. Softw. Eng.*, vol. PP, no. 99, p. 1, 2016.

[13] F. Křikava, P. Collet, and R. B. France, "ACTRESS: Domain-specific modeling of self-adaptive software architectures," in *Proceedings of the ACM Symposium on Applied Computing*, pp. 391–398, 2014.

[14] B. Kitchenham, D. Budgen, and P. Brereton, "Guidelines for performing systematic literature reviews in software engineering," *Int. Conf. Soft. Engin.*, vol. 45, no. 4ve, p. 1051, 2006.

[15] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele Univ.*, vol. 33, no. TR/SE-0401, p. 28, 2004.

[16] G. . Giachetti, F. Valverde, and B. Marín, "Interoperability for model-driven development: Current state and future challenges," in *Proceedings - International Conference on Research Challenges in Information Science*, 2012.

[17] M. Becker, M. Luckey, and S. Becker, "Model-driven performance engineering of self-adaptive systems: A survey," in *QoSA'12 - Proceedings of the 8th International ACM SIGSOFT Conference on the Quality of Software Architectures*, pp. 117–122, 2012.

[18] M. Petticrew and H. Roberts, *Systematic Reviews in the Social Sciences: A Practical Guide*. 2006.

[19] 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.*, vol. 80, no. 4, pp. 571–583, 2007.

Extending HL7 RIM Model to Capture Physical Activity Data

Rishi Saripalle
School of Information Technology, Illinois State University, Normal, IL
rishi.saripalle@ilstu.edu

***Abstract*—Experts recommend to include physical activity in everyone's daily routine as it helps to curb various medical issues. They provide diverse healthcare services (e.g., interventions, exercise routine, personal trainer plan, therapy, etc.) to promote healthy living and to increase physical activity in one's ever increasing hectic schedules. An individual generates a lot of data when he/she performs a physical activity. However, this data (exercises, training plans, etc.) and any other associated data (e.g., results, vitals, calories, weight, etc.) is not represented using commonly agreed standards. The lack of standards has given rise to two primary issues. First, interoperability issues don't allow the physical activity data to be shared and integrated with heterogeneous healthcare systems. Second, various organizations and research programs still majorly rely on paper to record the data. This makes it highly impractical to develop any data-driven physical activity applications or evidence-based program. This article provides a detailed analysis of the problem and provides a potential solution in terms of extending HL7 RIM model using health and life sciences entities from Schema.org.**

Keywords-physical activity model; exercise model; HL7 RIM physical activity; HL7 exercise; EIM; FHIR; FHIR exercise

I. INTRODUCTION

In general, Physical Activity (PA) is defined as any body movement that works your muscles and requires energy expenditure. It is proven that 30 minutes of daily physical activity can significantly reduce a variety of chronic diseases, improve mental health and overall well-being of an individual [1]. Even though the benefits of performing any type of physical activity on the individual's overall health has been proven, physical inactivity is still the fourth leading cause of death [2]. It also manifested in higher health care expenditures [3], [4]. To tackle physical inactivity, experts have designed numerous programs and interventions to increase physical activity among various age groups, irrespective of their medical condition [5]-[8]. The data obtained through these interventions/programs is recorded, analyzed and reported either using either paper (notes, journals, reports, etc.) or semi-structured digital (e.g., excel, text, etc.) format. Experts also recommend *exercise* - one the most common and popular way to achieve physical activity or to reach personal health goals. Exercises come in various forms and can vary from low-intensity exercises such as walking, strolling, sleeping, etc. to vigorous-intensity exercises such as running, swimming, sports, etc. Currently, most of this exercise data (e.g., the name of the exercise, repetitions, duration, process, etc.) generated by an individual/patient is recorded using paper or at best using a mobile app. For example, weight training is one the most popular exercises and individuals who include it their routine still use manuals (fitness journals or notebooks) to record their training/routine data. Further, in today's digital era, the vital sign data (heart rate, blood pressure, etc.) and other associated data (calories, step count, elevation, etc.) generated by an individual/patient during the exercise or any physical activity is recorded by wearables. The wearables constantly monitor the individual's vitals and synchronize the data to a mobile app which latter pushes the data to a data repository. Eventually, this process creates data silos[1] as the data is captured by a variety of apps, represented using proprietary formats, and stored in proprietary data hubs that might not communicate with one another or to a healthcare system. In parallel, individuals also use various paper and mobile apps (e.g., MyfitnessPal[2], CalorieKing[3], etc.) to record their calorie consumption, food type, and nutrient information for various healthcare reasons. Similar to wearables, these mobile apps capture the data using proprietary data formats and later synchronize it to a proprietary data hub.

So, *what is the problem here*? *First*, the physical activity data captured by these various instruments (interventions or exercise training or personal training, etc.) on an individual/patient is majorly paper-based or digital format without following any standards. This gives rise to *interoperability issues*, an issue previously encountered with paper-based health records, that doesn't allow the data to be exchanged between heterogeneous systems, specifically with an Electronic Health Record (EHR) or any standard clinical system. What is the impact? The healthcare expert(s) do not have a wholistic view of an individual/patient health. *Second*, due to data interoperability issues, experts will not be able to aggregate the data from multiple diverse sources to design evidence-based physical activity programs as shown in Figure 1. For instance, Rishi Saripalle age 30 with no serious medical condition approaches a trainer to be physical activity and achieve his goals. Most of the trainers use their knowledge and experience to design an activity routine to help Rishi reach his goal. If a diet is required,

[1] https://www.bedrockdata.com/blog/the-danger-of-data-silos-part-1-where-do-they-come-from

[2] https://www.myfitnesspal.com/

[3] http://www.calorieking.com/

DOI reference number: 10.18293/SEKE2017-53

Rishi might seek assistance from a nutritionist to put a plan together that works with his training. How can the trainer and the nutritionist prove that this training and diet plan works? Currently, the physical activity domain cannot provide evidence to such questions or questions with a similar theme. However, the domain of biomedical informatics can provide evidence to such questions as its medical data can be aggregated because the data is represented and shared using healthcare standards [9].

Figure 1: Interoperability issues with physical activity data.

These mentioned issues in the physical activity domain are primarily due to lack of *standards*, both structural and semantic, for *representing* and *sharing* physical activity data. This preliminary research is focused on structural standards. The idea is to use existing healthcare standards and other standard models/vocabulary to represent and share physical activity data across diverse healthcare systems. This goal is achieved by extending the HL7 Reference Information Model (RIM) [10] with health and life sciences entities developed by extending Schema.org model. Schema.org[4] is a collaborative community effort to create and maintain schema(s) for data represented on the internet. To the best of our knowledge, this is a novel effort to capture physical activity data by extending existing healthcare standards. The rest of the article is organized as follows. Section 2 will provide the reader detailed background knowledge on HL7 RIM and Schema.org. Section 3 provides the proposed design with detailed analysis, software model(s) and its application. Section 4 concludes the article with future research.

II. Background Information

The primary issue identified in the previous section is the lack of *common standard(s)* for representing and sharing physical activity data. The domain of biomedical and healthcare informatics faced similar crisis i.e. using paper and unstructured format for recording and sharing patient summaries, laboratory results, prescriptions, and other relevant data. The Electronic Health Record (EHR) was the viable solution and the medical community worked diligently to provide reliable and scalable standards for representing and sharing EHR. Various structural (e.g. Clinical Care Document (CCD), Clinical Document Architecture (CDA) [10], HL7 V2 and V3 messaging standards, etc.) and semantic (e.g., SNOMED-CT [11], ICD [12], etc.) standards are developed to provide a sound framework for supporting EHRs. The HL7 V2 standard was developed from the perspective of the clinicians and hospitals, but is not backed by strong software modeling principles. The HL7 V3 standard was designed using fundamental software modeling principles and formal object-oriented methodology primarily targeting biomedical informatics experts.

The HL7 Reference Information Model (RIM) is the cornerstone of the HL7 V3 messaging design and development. The RIM model has six core classes: Act, ActRelationship, Participation, Roles, RoleLinks, and Entity. Everything happening in the domain is an Act. The Act is further specialized to define other acts such as Medications, Procedures, Observations, etc. The ActRelationship connects two or more Acts using relations such as composition, precondition, revision, etc. Participation defines the context for an Act. The participants are assigned a Role such as a patient, provider, practitioner, specimen, employee, etc. Multiple roles involved in the Act are related using RoleLink. Finally, Role is assigned to Entity such as a person, organization, place, etc. In summary, HL7 V3 messaging standard is based on the HL7 Reference Information Model with six backbone classes with well-designed hierarchies and abstraction.

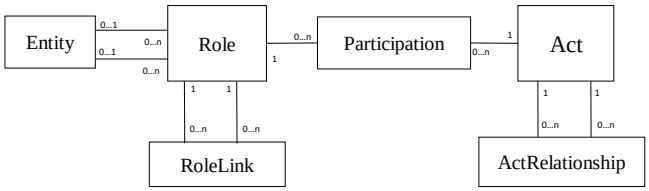


Figure 2: The HL7 Reference Information Model core classes.

Figure 3 shows a HL7 V3 message that captures "Heart Beat" (recorded as an observation (OBS)) as an outcome (OUTC) of the activity "Jogging" (also recorded as observation (OBS)).

```
<?xml version="1.0" encoding="UTF-8"?>
<entry typeCode="DRIV">
    <act classCode="ACT" moodeCode="EVN">
        <entryRelationship typeCode="SUBJ" inversionInd="false">
            <observation classCode="OBS" modeCode="EVN">
                <code code="1968006"
                codeSystem="2.16.840.1.113883.6.96"
                    displayName="Jogging" codeSystemName="SNOMEDCT"/>
                <entryRelationship typeCode="OUTC"
                                   inversionInd="true">
                <observation classCode="OBS" modeCode="EVN">
                    <code code="248646004" displayName="HeartBeat"/>
                        codeSystem="2.16.840.1.113883.6.96"
                    <statusCode code="Completed"/>
                    <effectiveTime value="20170301"/>
                    <value xsi:type="PQ" value="120" unit="bpm"/>
                </observation>
                </entryRelationship>
            </observation>
        </entryRelationship>
    </act>
</entry>
```

Figure 3: HL7 V3 message for representing heart beat due to jogging.

With the current standard, only a few physical activities and the associated data can be represented and shared. For example,

[4] http://schema.org/

activities such as jogging, running, swimming, etc. and any associated data (e.g., heart rate, blood pressure, calories, etc.) generated during the activity can also be captured using RIM. However, not all physical activities can be represented using the RIM such as weight training, exercise plans, diet, therapy, etc. However, the formal object-oriented model can be extended to capture data based on the requirement - physical activity data.

To further support this statement and the research goal, the exercise community is pursuing Exercise is Medicine (EIM) [13]. The primary goal of EIM is to establish physical activity as a requirement for patients that can be prescribed by healthcare practitioners and persist it in an EHR. An EIM[5] solution has three modules: Physical Activity Vital Sign (PAVS), EIM Prescription and EIM Referral. The PAVS records the physical activity levels of the patient by using the PAVS questionnaire. The EIM Prescription is the most interesting module which allows experts to *prescribe* a physical activity to the required patients similar to a medication prescription. Figure 4 shows the flow chart of embedding an EIM solution (three modules) into an EHR.

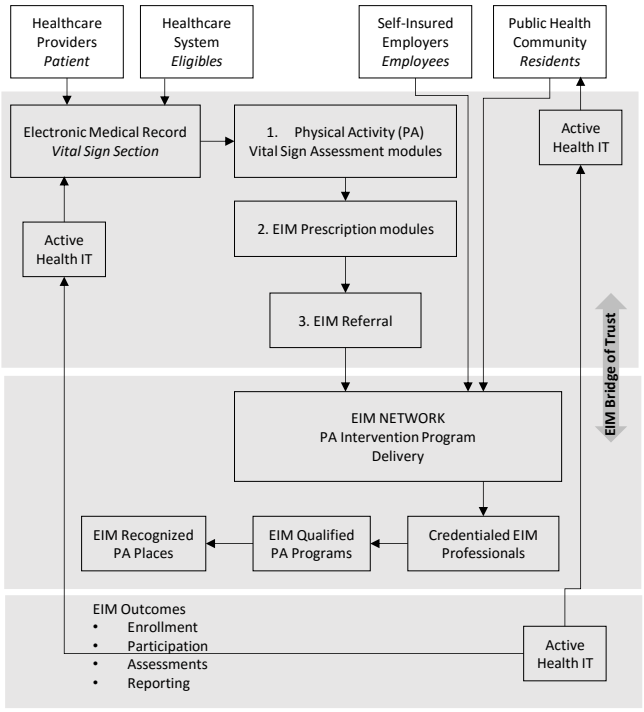


Figure 4: Exercise is Medicine in an EHR.

Apart from reusing the existing standards and community driven efforts such as EIM, there are only a handful of research studies that have identified and reported the need to capture physical activity data for persistence, longitudinal healthcare, and analysis. Kim et al., [14] have developed a physical activity semantic model that is comprised of activity keywords, qualifiers, response measures and context. These entities are obtained by reviewing 302 physical activity questions collected from standardized questionnaires and public data repositories. Sallies [15] identified the importance of physical activity similar to other experts and emphasized the need for the healthcare system to treat physical activity as a *vital sign*. This will manifest in recording and be observing the patient's physical levels during their medical visits. Coleman et al., [16] conducted a study to describe the facts and validity of an exercise vital sign (EVS) for use in the outpatient electronic medical records. After 18 months, 86% (1,537,798) of all eligible patients had an EVS in their electronic medical record, 36.3% of patients were completely inactive (0 min of exercise per week), 33.3% were insufficiently active (more than 0 but less than 150 minutes/week), and 30.4% were sufficiently active (150 min or more per week). Lobelo et al., [17] identified that mobile health (mHealth) applications and wearables can advance the assessment and integration of physical activity (PAVS module in EIM) in clinical settings and support community-based interventions. However, none of the discussed research studies leverage or extend an existing healthcare standard which is EHR compliant to capture the physical activity data.

Schema.org is a community effort to design standard schema's to provide structure to the data published on the internet. The Schema.org vocabulary includes entities, relationships between entities, and it can be extended through the Schema.org extension model. Millions of modern sites use Schema.org to markup their webpages so that they are machine readable and search engine friendly. Figure 5 shows the description of an article using ScholarlyArticle entity on the HMTL webpage. The ScholarlyArticle is the entity (itemtype) that can be reached at http://schema.org/ScholarlyArticle and the article is described using the ScholarlyArticle attributes (itemprop).

```
<div itemscope itemtype="http://schema.org/ScholarlyArticle">
    <span itemprop="author">Saripalle, Rishi</span>
    <span itemprop="name">UMLS Semantic Network as a UML Metamodel for
        improving Biomedical Ontology and Application Modeling
    </span>
    <div itemprop="isPartOf" itemscope
            itemtype="http://schema.org/PublicationIssue">
        <span itemscope itemtype="http://schema.org/Periodical"
                itemid="#periodical">
            <span itemprop="name">IJHISI</span>
        </span>
        <span itemprop="datePublished">2015</span>
        <span itemprop="isPartOf" itemscope
                itemtype="http://schema.org/PublicationVolume">
        <span itemprop="volumeNumber">10</span>
        </span>
    </div>
    <div itemprop="isPartOf" itemscope
            itemtype="http://schema.org/PublicationIssue">
        <span itemprop="issueNumber">2</span>
    </div>
</div>
```

Figure 5: Representing an article using ScholarlyArticle entity in Schema.org.

This research will be using the Health and Lifesciences[6], a schema designed by extending the core Schema.org vocabulary that captures medical vocabulary such as Drug, Condition, Causes, Exercise, Anatomy, Device, etc.

[5] http://www.exerciseismedicine.org/

[6] http://health-lifesci.schema.org/

III. Extending Rference Information Model with Entities from Schema.org

In this section will be present and discuss in detail our proposed extension to Reference Information Model (RIM) using entities from Health and Lifesciences vocabulary - an extension to core Schmea.org model. The entities that will be adapted into RIM model are: PhysicalActivity, ExercisePlan and Diet. Figure 6 shows the hierarchy from Schema.org, where ExercisePlan *isa* PhysicalActivity which in turn *isa* MedicalEntity (part of Health and Lifesciences vocabulary) which in turn *isa* Thing (a top concept in core Schema.org schema).

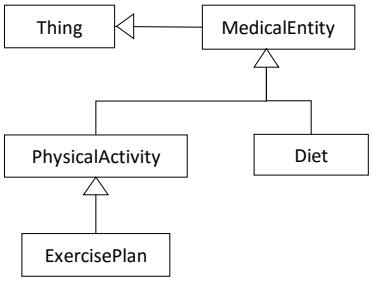


Figure 6: The hierarchical relationship between PhysicalActivity, ExercisePlan and Diet in Schema.org vocabulary.

As previously mentioned, Act in a RIM class that captures everything that happens and is extended to define other classes (Procedure, Observation, Supply, etc.). The Act is extended with the three aforementioned entities from Schema.org. It must be noted that the Diet class from RIM is merged with Diet entity from Schema.org to enrich the semantic expressiveness of the class. Figure 7 shows the extended RIM model with the Schema.org entities (in gray).

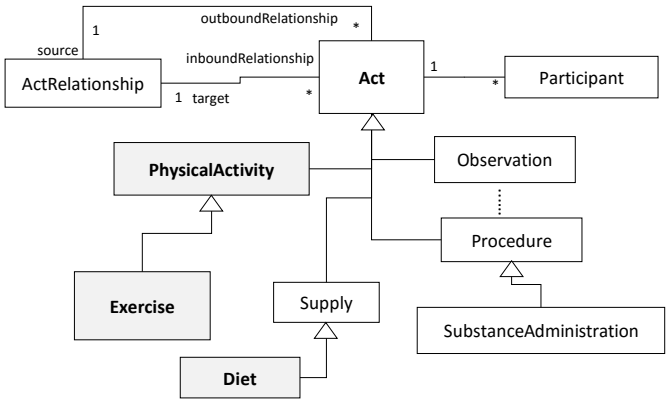


Figure 7: Extended RIM Model with Schema.org entities

As shown in Fig 7, PhysicalActivity and Exercise (renamed from ExercisePlan) are the subclasses of Act (similar to Observation) with "PHY" and "EX" as the respective classCode. The Diet entity from Schema.org is merged with the Diet class in RIM which inherits from Supply. The Schema.org entities adapted into the RIM model have their own properties and also the properties inherited from their parents, i.e. MedicalEntity and Thing (Figure 6). When these entities are defined as the subclasses of Act, they inherit the attributes and relationships of Act. Table 1 shows in detail the properties of the PhysicalActivity (with datatypes) from Schema.org and any equivalents attributes (with datatypes) PhysicalActivity class inherits from Act. Few properties of PhysicalActivity entity can be expressed as relationships with other RIM classes (Role, Entity, etc.) and a few properties can be eliminated as we are translating a Schema.org entity that is primarily used in describing data on the internet to a modeling class. The properties (expressed as "N/A" in Table 1) with no equivalent match in the Act or related to other RIM classes are defined as the attributes of PhysicalActivity class as shown in Figure 8.

TABLE 1: PhysicalActivity properties from Schema.org and equivalent attributes (relationships) in RIM model.

Property (from Schema.org)	**Datatype (from Schema.org)**	**Equivalent Attribute from Act Class**	**Equivalent HL 7 Datatype**
associatedAnatomy	AnatomicalStructure AnatomicalSystem/ SuperficialAnatomy	N/A	CD (Coded Descriptor)
category	Text	N/A	ST (String)
pathophysiology	Text	N/A	ST
code	MedicalCode	Code	CD
guideline	MedicalGuideline	N/A	ST
alternateName	Text	N/A	ED (Encapsulated Data)
name	Text	Title	ED
description	Text	Text	ED
epidemiology	Text	TX	
Property (from Schema.org)	**Datatype (from Schema.org)**	**Expressed as a relationship with a RIM Class**	
legalStatus	MedicalGuideline	SubstanceAdministration or ST (String)	
medicineSystem	MedicineSystem	Entity	
recognizingAuthority	Organization	Organization	

The Exercise class also inherits attributes from Act as shown in Fig 7. Table 2 only shows the properties from Schema.org that will be translated as attributes in Exercise with their respective datatypes as shown in Figure 8. Table 3 shows the properties of the Diet (with respective datatypes) from Schema.org that are included in the Diet class as attributes. Figure 8 shows the complete Diet class in the extended RIM model. With the extended RIM model, the physical activity data can be shared via HL7 V3 messages. Figure 9 shows a simple HL7 V3 message that captures a new (represented using status) physical activity that affects the body composition (associatedAnatomy) of the patient and is authored by Dr. Rishi Saripalle. With availability to translate HL7 V3 to HL7 V2 messages, the message in Figure 9 can also be converted into HL V2 message, a more popular format which clinical information systems. With the extended model, an authorized organization or personal can prescribe a physical activity to a

patient, similar to a medication and the results can be shared with the prescribed party – one of the EIM goals.

TABLE 2: Exercise properties from Schema.org and equivalent attributes datatypes in RIM model

Property (from Schema.org)	Datatype (from Schema.org)	Equivalent HL 7 Datatype
activityDuration	Duration/ QualitativeValue	PQ (Physical Quantity)
activityFrequency	QualitativeValue/ Text	PQ
additionalVariable	Text	ST
exerciseType	Text	CD
intensity	QualitativeValue/ Text	PQ
alternateName	Text	ED
repetitions	QualitativeValue/ Number	QTY (Quantity)
restPeriods	QualitativeValue/ Number	QTY
Workload	QualitativeValue/ Energy	PQ

TABLE 3: Diet properties from Schema.org and equivalent attributes (relationships) datatypes in as RIM model.

Property (from Schema.org)	Datatype (from Schema.org)	Equivalent HL 7 Datatype
dietFeatures	Text	ST
physiologicalBenefits	Text	ST
risks	Text	ST
Property (from Schema.org)	**Datatype (from Schema.org)**	**Expressed as a relationship with a RIM Class**
endroders	Organization/Person	Organization/Person

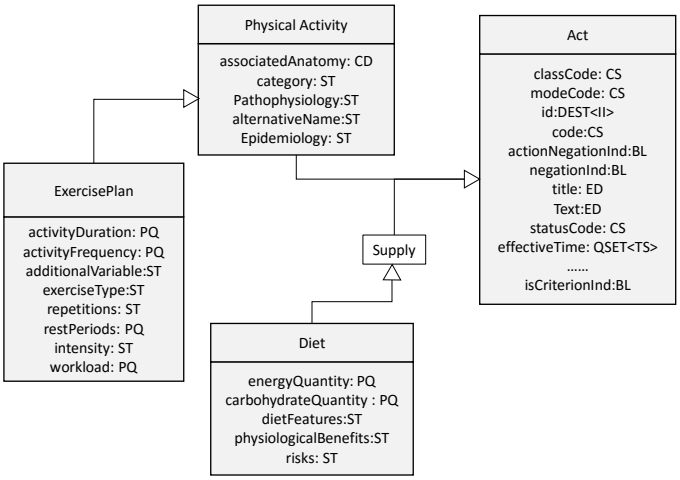


Figure 8: PhysicalActivity, Exercise and Diet classes in extended RIM model.

Figure 10 shows another HL7 message that captures a physical activity with a defined exercise plan, and an associated diet authored by an expert associated with an organization. The extended RIM model can also capture multiple exercise plans that can be associated with a physical activity. In the course of this research, the author has noticed that the physical activity/exercise/cardio domain doesn't have any standard terminology or ontology that express the vocabulary of the domain. Few existing medical standards (e.g., SNOMED) capture generic vocabulary such as jogging, walking, running, etc., but don't capture in-depth vocabulary such as weight lifting, chest press, bench press, plank, lunges, etc. The domain would require a standard vocabulary for unambiguous interpretation of shared activity data.

```
<entry>
     <physicalactivity classCode="PHY" modeCode="RQO">
          <code code="1968006" codeSystem="2.16.840.1.113883.6.96"
               displayName="Jogging" codeSystemName="SNOMEDCT"/>
          <title mediaType="text/html" data="Daily Activity"/>
          <associatedAnatomy code="M0002741" codeSystem=
               "2.16.840.1.113883.6.177" displayName=
                    "Body Composition" codeSystemName="MeSH"/>
          <statusCode code="new"/>
          <author/>
               <assignedAuthor>
                    <assignedPerson>
                         <name>
                              <prefix>Dr.</prefix>
                              <given>Rishi</prefix>
                              <family>Saripalle</prefix>
                         </name
                    </assignedPerson>
               </assignedAuthor>
          </author>
      </physicalactivity>
</entry>
```

Figure 9: A HL7 V3 message capturing physical activity

```
<entry>
     <physicalActivity classCode="PHY" modeCode="RQO">
          <code code="1968006" codeSystem="2.16.840.1.113883.6.96"
               displayName="Jogging" codeSystemName="SNOMEDCT"/>
          <title mediaType="text/html" data="Daily Activity"/>
          <associatedAnatomy code="M0002741" codeSystem="…" displayName=
                    "Body Composition" codeSystemName="MeSH"/>
          <statusCode code="new"/>
     </physicalActivity>
     <entryRelationship typeCode="COMP" inversionInd="false">
          <exercise classCode="EXP" modeCode="RQO">
               <code code="282961006" codeSystem=" . …" displayName="Squat"
                    codeSystemName="SNOMEDCT"/>
               <activityDuration xsi:type="PQ" value='3' unit=
                    "minutes"/>
               <repetitions xsi:type="QTY" value='3'/>
               <restPeriods xsi:type="PQ" value='1' unit="minute"/>
               <statusCode code="new"/>
      </exercise>
     </entryRelationship>
     <entryRelationship typeCode="COMP" inversionInd="false">
      <exercise classCode="EXP" modeCode="RQO">
          <code code="M0022909" codeSystem=". …" displayName="Weight Lifting"
               codeSystemName="MeSH"/>
          <activityDuration xsi:type="PQ" value='3' unit= "minutes"/>
          <additionalVariables mediaType="text/html" data="85% of
               your max and rest for 2 min after"/>
          <repetitions xsi:type="QTY" value='5'/>
          <restPeriods xsi:type="PQ" value='2' unit= "minute"/>
          <statusCode code="new"/>
      </exercise>
    </entryRelationship>
    <entryRelationship typeCode="COMP" inversionInd="false">
      <Diet modeCode="RQO">
          <energyQuantity xsi:type="PQ" value='2000' unit= "calorie"/>
          <carbohydrateQuantity xsi:type="PQ" value='300' unit= "calorie"/>
          <statusCode code="new"/>
      </Diet>
      <author>
          <assignedPerson>
               <name>
                    <prefix>Dr.</prefix>
                    <given>Nutrition</prefix>
                    <family>Expert</prefix>
               </name>
          </assignedPerson>
          <representedOrganization>
               <name>ABC Experts</name>
               <addr nullFlavor="UNK"/>
          </representedOrganization>
      </author>
    </entryRelationship>
</entry>
```

Figure 10: A HL7 V3 message capturing physical activity, an exercise plan, Diet and associated author of the Diet.

IV. Conclusion and Future Work

In this research article, existing medical standard(s) which is compliant with EHR is extended to represent and share physical activity data across heterogeneous healthcare systems. This

goal is achieved by extending HL7 Reference Information Model (RIM) with entities defined in Health and Life Science vocabulary from Schema.org. To the best of our knowledge, this is a novel approach to represent the physical activity/exercise data using HL7 RIM and sharing it using HL7 V3/V2 messaging standards. To support this approach, in Section 1, the article presented the current situation in the domain and the immediate need for capturing the physical activity data (Fig 1). Section 2 provides the required background knowledge and other scholarly research efforts aligned with the proposed idea, especially EIM solution (Figure 4). Finally, the article demonstrates the capabilities of the extended HL7 RIM by constructing HL7 V3 messages (Fig 9 & Fig 10) using simulated data.

The work presented here is a very crucial first step towards capturing physical activity data and moving towards evidence-based practice, similar to biomedical and healthcare informatics. However, more work is required to expand, implement and integrate the proposed model into an EHR. Apart from Schema.org entities, integrating the paper-based format followed by various fitness organizations (e.g., LA Fitness, Gold gym, etc.) and expert's knowledge would add value to the extended model. We are continuing our effort to implement the designed model using openEHR as a proof of concept. Further, with increasing use of mobile apps, ubiquitous computing for capturing data and the influence of light-weight web services, the proposed extended model classes will also be translated into Fast Healthcare Interoperability Resources (FHIR)[7] resources [18]. This is achieved by using HAPI[8], an open source implementation of FHIR standard. With the integration of the physical activity data into an EHR, an expert(s) will have a *wholistic* view of the patient health.

REFERENCES

[1] Anonymous (2016, June 6). *Physical Activity Guidelines for Americans*.

[2] K. Harold *et al*, "The pandemic of physical inactivity: global action for public health," *The Lancet*, vol. 380, pp. 294, 2012.

[3] S. A. Carlson *et al*, "Inadequate Physical Activity and Health Care Expenditures in the United States," *Prog. Cardiovasc. Dis.*, vol. 57, pp. 315-323, 2016/05, 2015.

[4] J. F. Bell *et al*, "Health-Care Expenditures of Overweight and Obese Males and Females in the Medical Expenditures Panel Survey by Age Cohort," *Obesity*, vol. 19, pp. 228-232, 2011.

[5] A. Carlin, M. H. Murphy and A. M. Gallagher, "Current influences and approaches to promote future physical activity in 11--13 year olds: a focus group study," *BMC Public Health*, vol. 15, pp. 1-12, 2015.

[6] M. Temple and J. C. Robinson, "A systematic review of interventions to promote physical activity in the preschool setting," *Journal for Specialists in Pediatric Nursing*, vol. 19, pp. 274-284, 2014.

[7] C. Amaya-Castellanos *et al*, "Development of an educational intervention to promote healthy eating and physical activity in Mexican school-age children," *Eval. Program Plann.*, vol. 52, pp. 159, 2015.

[8] B. Belza *et al*, "From research to practice: EnhanceFitness, an innovative community-based senior exercise program," *Topics in Geriatric Rehabilitation*, vol. 26, pp. 299-309, 2010.

[9] M. Blechner, R. Saripalle and S. Demurjian, "A proposed star schema and extraction process to enhance the collection of contextual amp; semantic information for clinical research data warehouses," in *2012 IEEE International Conference on Bioinformatics and Biomedicine Workshops*, 2012, pp. 798-805.

[10] K. W. Boone, *The CDA™ book*. Springer, 2011.

[11] T. Benson and G. Grieve, *Principles of Health Interoperability: SNOMED CT, HL7 and FHIR*. Springer, 2016.

[12] (2010). *International Classification of Diseases*. Available: http://www.who.int/classifications/icd/en/.

[13] F. Lobelo, M. Stoutenberg and A. Hutber, "The Exercise is Medicine Global Health Initiative: a 2014 update," *British Journal of Sports Medicine*, vol. 48, pp. 1627, 2014.

[14] H. Kim *et al*, "Developing a Semantic Model to Describe Physical Activity Data," *Stud. Health Technol. Inform.*, vol. 225, pp. 447-451, 2016.

[15] R. Sallis, "Developing healthcare systems to support exercise: exercise as the fifth vital sign," *British Journal of Sports Medicine*, vol. 45, pp. 473, 2011.

[16] K. J. Coleman *et al*, "Initial validation of an exercise "vital sign" in electronic medical records." *Medicine & Science in Sports & Exercise*, vol. 44, 2012.

[17] F. Lobelo *et al*, "The Wild Wild West: A Framework to Integrate mHealth Software Applications and Wearables to Support Physical Activity Assessment, Counseling and Interventions for Cardiovascular Disease Risk Reduction," *Prog. Cardiovasc. Dis.*, vol. 58, pp. 584-594, 2016/05, 2016.

[18] M. Anwar and C. Doss, "Lighting the Mobile Information FHIR: How FHIRframe Could Dramatically Improve Mobile Health and Change HIM in the Process," *Journal of AHIMA*, vol. 86, 2015.

[7] http://hl7.org/fhir/resourcelist.html

[8] http://hapifhir.io/

An Approach to Mobile Application Testing Based on Natural Language Scripting

Chuanqi Tao[①], Jerry Gao[②③], Tiexin Wang[①]

① College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

②Department of Computer Engineering , San Jose State University, USA

③Taiyuan University of Technology, Taiyuan, China

Correspondence to: jerry.gao@sjsu.edu

Abstract: With the rapid advance of mobile computing technology and wireless networking, there is a significant increase of mobile subscriptions. This brings new business requirements and demands in mobile software testing, and causes new issues and challenges in mobile testing and automation. As there are multiple platforms for diverse devices, engineers suffer from the different scripting languages to write platform-specific test scripts. In addition, a unified automation infrastructure is not offered with the existing test platform. This paper proposes a novel approach to mobile application testing based on natural language scripting. A Java-based test script generation approach is developed to support executable test script generation based on the given natural language-based mobile app test operation scripts. A prototype tool is implemented based on some open sources. Finally, the paper reports empirical studies to indicate the feasibility and effectiveness of the proposed approach.

KEYWORDS: mobile application testing; test automation; scenario-based testing; behavior-based testing

1. Introduction

Today mobile computing is the beginning of another revolution that is going to sweep the consumer industry with its fast paced growth. The demand for smart phones and tablets coupled with their expensive data plans is already soaring high and is a trillion dollar industry. As mobile APP and mobile application vendors, they have encountered the following critical issues. Testing mobile apps on different mobile platforms and various devices becomes very costly and tedious due to the fast upgrading of mobile devices, and rapid updates of mobile platforms and technologies. Testing mobile web applications on diverse mobile browsers on different devices become very difficult and costly due to increasing mobile scalability, constantly updates of mobile devices and mobile technologies, diversity of involved mobile web technology, and fast upgrading mobile application services [1, 2, 3, 18].

Most of the present research work focuses on providing solutions to the technical problems in the following areas: a) mobile app white-box testing methods; b) GUI-based test technique for mobile apps [5]; c) usability testing [6]; d) ad hoc scripting test tools; e) wireless network testing [9]. There are two challenges in mobile testing: the first issue is too costly to deal with diversity of mobile test environments on varieties of mobile devices, and the other is the lack of cost-effective method and platforms to support a unified mobile test automation crossing different mobile platforms on diverse devices. Most of the existing mobile testing tools support GUI-based functional testing and few support load/performance testing. Currently, a unified automation infrastructure is not offered with the existing test tools. In addition, there is lack of well-defined mobile test scripting method to deal with the massive multiple mobile test running. Therefore, test automation central control is needed to support behavior-based testing or scenario-based testing at multiple levels.

In this paper we develop practical solutions and design a unified mobile test platform to address the needs and limitations. The intent is to develop a unified system that enables test engineers to dynamically test the mobile applications without the dependency on any scripting language. The proposed approach will enable test engineers to define the scenarios to be tested in plain natural language that supports seamless and efficient testing of mobile applications. In this approach, we attempt to design a system capable of testing the properties of the application automatically once the scenarios are written for a set of features from natural language. As the test cases can be purely user written stories, we can simple use natural language to write the functionality and scenarios to test, and then script the code to perform those tests automatically. Further, the code to execute these tests would be a part of step-definitions in the overall test automation framework.

The paper is structured as follows. The next section presents the background and related work. The architecture of the BDD-based natural language processing analysis is discussed and the designed and implemented prototype tool for the proposed approach is presented in Section 3. Case studies of testing on several mobile apps are shown in Section 4. Conclusion and future work are summarized in Section 5.

2. Background and Related Work

2.1 Background

Behavior driven development (BDD) includes a work flow which requires the software developers to test the behavior

DOI reference number: 10.18293/SEKE2017-170

of a piece of code as and when it is developed [21, 23]. It is used by the stakeholders to verify if the product approach is correct or wrong. BDD is based on scenarios to check the validity of the system. Using this design flow based on the scenarios will allow us to analyze the whole piece of code, scenario by scenario. This not only helps developers understand the functioning of the component, the stakeholders can also verify the step process. BDD is written in natural language which is easier for both the parties to understand and verify the functionality. Natural language allows the developers to see if there is any ambiguity in the thought process of a system due to a complicated process. BDD works in a sentence by sentence approach rather than taking the whole code as a whole. It allows everyone to see which line is getting the correct result and which is not. As the test cases can be purely user written stories, there is no need to write test cases in code. Therefore, we can use natural language to write the functionalities and scenarios to test, and then script the code to perform those tests automatically. Thus, we can easily write human understandable user stories and this can enable even beginners to test the applications. Further, the code to execute these tests would be a part of step-definitions. This acts as a layer of abstraction and promotes a series of advantages and efficiencies in mobile application testing.

2.2 Related Work

Up to today, many papers have been published to address different testing areas in mobile applications. These areas include such as white-box and black-box testing, quality-of -service testing, wireless connectivity testing, and test automation frameworks.

Existing white-box testing methods are still applicable to mobile apps. For example, Java Pathfinder [1] is a mobile program verification tool supporting white-box mobile Java program testing. Engineers can use this tool to detect race conditions and deadlocks based on UML state machines and symbolic execution. Mahmood and his colleagues [2] use a white-box approach to generate test cases with two program-based models (call graph and architectural) to achieve mobile program code coverage. Many black-box testing techniques are useful in mobile apptesting. Random testing and the scenario-based testing method [3] are good examples. GUI-based testing has been discussed in numerous papers. For instance, Anand and his colleagues [4] introduced an automated testing approach to validating mobile GUI event sequences for smart-phone apps. Similarly, Amalfitano and his colleagues [5] presented a tool AndroidRipper that uses an automated GUI-based technique to test Android apps in a structured manner.

There are many tools that support BDD such as cucumber, jBehave, twist etc. In recent times, a large number of mobile testing tools have been developed to support mobile development [17, 21, 23]. As the number of mobile products is increasing, the testing of these products is important. We need to choose the testing tool that best suits the product. Thereby, increasing demand for mobile devices has lead to the needs for developing tools to test them at a high level. Based on our research, we found that most of the testing tools either perform the GUI testing or load and performance testing. But all the tools have some limitations. For instance, many tools require engineers to learn different scripting languages in order to write platform specific and tool specific scripts which would be difficult for them. However, engineer still need new test models to address special needs in testing mobile applications.

Unlike the existing research, our goal in this paper is to address the growing needs of the mobile market like the use of natural language to test the mobile applications. Our approach aims in providing solution to such problems and our research has a wide impact on the market which keeps on changing rapidly.

3. BDD-Based Natural Language Scripting for Mobile Application Test Generation

The order of behavior driven development in the form of acceptance tests is as follows.

(a) A scenario is described to define the action or behavior of the system to be tested;
(b) Step definition is the part where the natural language is converted into the actual working code based on the mapping of the constructs of the natural language. A specific regular expression is used to determine the code which is to be executed on reading the sentence;
(c) A code skeleton is needed to comply with the tool that is being used, so that efficient codes can be written and run;
(d) Codes in the code skeleton are then run which will show us the results. This will verify if the code has passed or failed depending on the scenario.

We used the tool *cucumber* (*https://cucumber.io/to*) to run the nature language scripts written in *Gherkin* language. The server side code i.e., the step definitions for *cucumber* are written in Ruby language which contains all the definitions and classes needed to test the functionalities of the mobile app. Cucumber runs and points out the steps which have been used but not yet defined and prompts the user to define each of those steps. These step definitions can be written in any language where we used Ruby for defining steps in our approach. The framework Calabash (*http://calabash.sh*) is used to support automated tests written in ruby and Gherkin. The framework contains all the scenarios in the feature file written in nature language, Gherkin. It also contains the step definitions which act as a server side code written in Ruby language. A number of self-defined functions are used in our current approach.

In brief, the process of BDD for mobile testing here includes scenario description, step definition, code skeleton, and implementation. Figure 1(a), (b), (c), and (d) shows the

DOI reference number: 10.18293/SEKE2017-170

examples of BDD in the scenario of *telephone call*.

Scenario: *Placing a call*

* Ada picks up the receiver from the telephone
* She dials the number 6-345-789
* The telephone places a call

(a) Scenario

```
Given /^Ada picks up the receiver from the telephone$/ do
  @telephone = Telephone.new
  @receiver = @telephone.pickUp
end
```

(b) Step Definition

```
class Telephone
  def pickUp
  end
end

class Receiver
end
```

(c) Code Skeleton

```
class Telephone
  attr_reader :receiver

  def initialize
    @receiver = Receiver.new
  end

  def pickUp
    @receiver
  end
end
```

(d) Implementation

Figure 1 A sample BDD process

3.1 Modeling Semantic Relation of Natural Language

In order to obtain a semantic relation between the keywords of the defined language, we need to put together the sentence in a syntactical manner. Both these parsers will help us create a Phrase Structure tree (PST). This phrase structure tree will be the backbone of our natural language. Our approach will try and break all the keywords to be used into skeleton codes. We firstly divide the language into grammatical components, and then define each and every grammatical component. Secondly, a class and function is tied to every component. Thirdly, we define the code for each and every specific tie up. Finally, we run the scripts to return the behavior. Figure 2 shows some examples on how to break a sentence into a PST. Figure 3 presents some key words in our language (bold words). When a part of natural language has been written, a parser is required to break the language into components and relate with the skeleton code in the background.

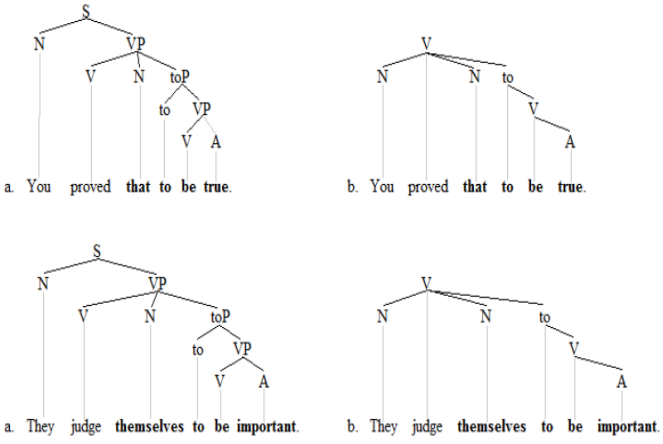


Figure 2 PST examples

```
Feature: Multiple Scenario
In order to extend cucumber functionality
as a project developer
I want to include multiple test scenarios

Background: Given are a set of scenarios in a hierarchical order

@user @app @scenario @environment

Scenario Outline: Check network connectivity
And check another scenario for security certificates
And <another scenario after that>

Given security certificates are exchanged
And <another scenario is completed>
And network connectivity checks out

When connection is made by the user
And connection is fully authenticated

Then run the Selenium tests
And Appium tests
And Return the results in the console
```

Figure 3 A sample keywords in our language

3.2 Design and Implementation for the Approach

This section presents the developed software system architecture and explains about the involved components and their relations and connectivity.

Figure 4 shows the workflow of the designed system. The blue colored components are developed in our approach. Most of the yellow colored components can be built from the existing work and some open source tools. The Gherkin extension is implemented where the majority of the functionality and logic is defined. The intelligence goes in extending the behavior of Cucumber by defining extensions based on scenarios written in nature language. The part enclosed in the dotted lines is the work flow of test scripting in the solution. Our approach aims to design a prototype systematic tool which can support to generate mobile application testing scripts based on the requirements of the user input. The user input determines which script to be selected from the database and which environment to set up. We analyze the user inputs and categorize them into three different parts. Then these parts will be segregated into the Gherkin extension which forms the scripts based on the user inputs. Next, some implemented key components and used technology are discussed in details.

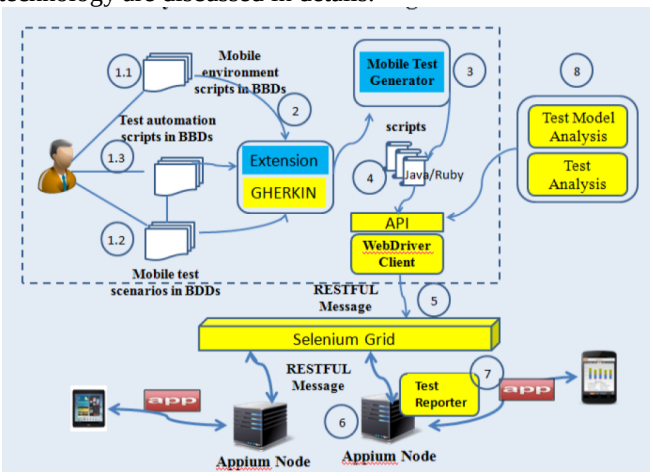


Figure 4 The workflow of the designed system for mobile application testing

DOI reference number: 10.18293/SEKE2017-170

User inputs - Three types of user inputs are shown in Figure 4. The first set of inputs refers to the environment parameters. These set of options is predefined by the system user and would be allowed to choose the options according to the test suite or the test script. Test automation scripts in behavior driven development will be supplied by the user for automating test scripts execution. These scripts are being handled by some other team. The third input would be the required test scenarios for testing the application. The test scenarios are implemented at a multiple level.The existing work only focuses on one level scenario that was being executed. In this approach, we try and implement the scenarios at multiple levels.

Gherkin extension - Gherkin is a simple natural language based programming language. It does not have a very complex and detailed syntax. Only a few keywords are required to use gherkin as a language. The input supplied by the user is processed by the Gherkin extension. When we run the gherkin scripts in cucumber, it generates a report based on the keywords and sentences provided no matter the script will run or not. If it runs, then we check if it behaves in the way we have defined in gherkin script. After that, the related information is sent to the Mobile Test generator (the component shown in Figure 4) for execution.

Mobile test generator - In this mobile test generator, all the additional functionality and logic is applied to define the test scripts. The Cucumber understands the natural language based scripts, while internally uses Java/ Ruby language for procession. The interactions among the servers happen by using the defined API's.

Defining Steps in Cucumber- We used the tool cucumber to run our natural language scripts written in Gherkin language. The server side code i.e., the step definitions for Cucumber are written in Ruby language which contains all the definitions and classes needed to test the functionalities of the mobile app. The sample features and scenarios are illustrated as follows.

Cucumber Components- Features and Scenarios - Following this approach for defining the steps will make the steps/scenarios more complete and readable though the order is not critical. Cucumber runs and points out the steps which have been used but not yet defined and prompts the user to define each of those steps. These step definitions can be written in any language where we used Ruby for defining steps in our approach. Even if one step is pending, the entire scenario is marked as pending. In case the first step fails to execute or is pending, the entire scenario is skipped. Step definitions can be used and re-used and one step can be called from another.

We have used the framework Calabash to support automated tests written in ruby and Gherkin. It contains all the scenarios in the featurefile written in natural language, Gherkin. It also contains the step definitions which act as a server side code which are written in Ruby language. Calabash Android provides the client-server architecture. We have included many custom defined functions which are used in our current approach.

4. Case Studies

To apply the developed tool for dynamic mobile application testing, we used several realistic mobile applications and system. NDTV, Waze, and WordPress APP are selected for the study objects. We created step definitions for multiple apps to implement different functionalities and scenarios. The operational flow to set up the calabash android and finally run the test scripts is as below.

Step 1: To define the calabash android environment;
Step 2: To set up the *apk* parameters to build;
Step3: To obtain the *apk* and start the calabash console;
Step 4: To generate a feature directory;
Step 5: To run cucumber;
Step 6: Run the calabash android;
Step7: Calabash Android Dem Spec Created.

4.1 Study Results

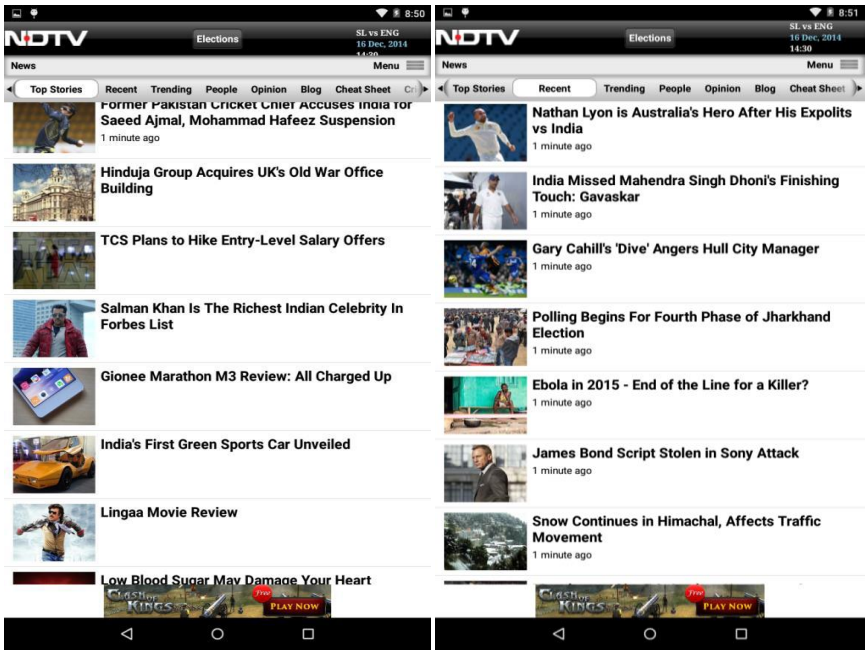
Case 1: NDTV App- Syntax

Initially we started working on the NDTV new app for testing the buttons and scrolling features of the app.

We defined multiple scenarios where a test engineer can test different properties according to the requirements. Here we explicitly defined 5 different scenarios. In the first scenario we change the headline once the app is launched. When the “news” shows, we define the steps to scroll down. As recent headlines appear, we further scroll down and use the button press feature to select the “recent” button on the app. Then we wait for progress and again scroll down. Yet again, we press the “trending” button and scroll down. In this way, we can create multi-level scenarios for testing different attributes of the application. We have used the NDTV *apk* file from the Google Play and tested it using the scenarios. When we try to run the calabash android gen command, it creates a sub directory.

When all the cases are tested successfully, it displays the results as passed including the number of successful scenarios. The scenarios are executed on an android device and the screenshots depict the execution summary for different scenarios for NDTV app. Figure 5 presents the test results when running the scripts for the app. Right below NDTV we can see the text “News” which will be read by the application as a starter sign to execute next steps in the scenario shown in Figure 5(a). The next steps are directing the application to scroll down the page three times. The second screenshot shown in Figure 5(b) here shows the same feature of the script file but a different scenario. Here the second tab after “Top Stories” is clicked which can be seen down below as “Recent”.

DOI reference number: 10.18293/SEKE2017-170



(a) (b)

Figure 5 Test results when running the scripts for 'NDTV' app

Case2: WAZE App- User Sign in Syntax

We further enhanced the functionalities of our approach and implemented user *sign-in* feature for waze app as depicted below. The scenario is to log into the application. The scenario is defined as below.

The test engineer logs into the application and upon login should see the User Agreement and then touch the Accept check box. Next user touches the Login button and enters the user id and password and then presses sign-in. Figure 6 shows the sample of syntax for testing the scenarios.

my_first-6.feature

```
Feature: To test a map based android app

Scenario:  To enter the app


Given I wait for 3 seconds
And I see the text "OUR AGREEMENT"
When I scroll down
And I scroll down
And I scroll down
Then I press view with id "settingsCheckboxText"
And I touch the "Accept" text

And I see the text "Welcome to Waze!"
And I touch the "Login" text
And I touch the "Existing users sign in:" text
And I enter text "lpshikhar" into field with id "userName"
And I enter text "carboncs15" into field with id "password"
And I press view with id "signin"
```

Figure 6 A sample syntax for testing the scenarios

Case 3: Wordpress App: Scenarios and Syntax

The next step was to execute some key functionalities for WordPress App. The below execution summary depicts the different scenarios for this app. Two scenarios are defined in detail. The first scenario is to log into the app being a valid user. The test engineer can validate the sign-in feature of app using this natural language based scenario. As the user gets to sign-in, it enters the username and password and then presses the sign-in button. As the user enters the application, it should see the "Posts" tab and press view for "more options". The user then logs out of the application. There is a two second timer after which the user is prompted to sign-in again.

The second scenario describes the scenario when the login information is incorrect. The user should get a message that "login information is not correct". Figure 7 presents the feature of two scenarios.

my_first-3.feature

```
Feature: Login feature

Scenario: As a valid user I can log into my app
    When I see "Sign in"
    Then I enter text "lpshikhar" into field with id "nux_username"
    And I enter text "shikharrawat" into field with id "nux_password"
    Then I press "Sign in"
    And I should see "Posts"
    And I press view with content description "More options"
    And I touch the "Sign out" text
    And I press the "Sign out" button
    Then I wait for 2 seconds
    And I should see "Sign in"
    Then I enter text "lpshikhar" into field with id "nux_username"
    And I enter text "shikharrawat" into field with id "nux_password"
    And take picture

Scenario: Wrong log in info
    When I see "Sign in"
    Then I enter text "lpshikhar" into field with id "nux_username"
    And I enter text "shikhar" into field with id "nux_password"
    Then I press "Sign in"
    And I should see "The username or password you entered is incorrect"
```

Figure 7 A sample syntax for testing the scenarios

4.2 Study Limitations

While designing the system and developing the tool, we faced several issues. Some of those issues are listed below.

We had used emulator for testing the apps which took

DOI reference number: 10.18293/SEKE2017-170

longer time to run the whole scenario. To improve this we replaced the emulator with the android tablet we have and reduced the running time. Regarding to the resign tool, when the apps are tested, the *.apk* file needs to use the same signature in our approach. To achieve this we used the *re-sign* jar which is a drag and drop java tool that strips the *apk* file from its signature and then signs it with our android debug key and solves the issue.

5 Conclusions and Future Work

As more constructions and deployments of mobile APPs and web applications on devices, engineers need more quality validation research and test automation tools to deal with the related issues and challenges. Currently, there is a need for an autonomous test scripting architecture for mobile apps, which we focus in this paper. This paper is a step forward towards natural language scripting. The intent of this work is to limit the dependency on any one scripting language and come out with the universal widely defined natural language.

The proposed approach has a lot of application scope in the future industry. There is also a scope for automation in the future as the present server side code requires manual testing. This approach suffers from some limitations. For example, the syntax is currently dependent on variables. Thus if the variables associated change, the step definitions need to be modified. The new id's need to be extracted from the app properties which is a complex problem in the known space. Therefore, there is tremendous scope to automate and reduce this dependency on the variables in the future work.

References

[1] H. van der Merwe et al., "Verifying Android Applications Using JavaPathFinder," ACM SIGSOFT Software Eng. Notes, vol. 37, no. 6, 2012,pp. 1–5.

[2] R. Mahmood et al., "A Whitebox Approach for Automated SecurityTesting of Android Applications on the Cloud," Proc. 7th Int'l WorkshopAutomation of Software Test (AST 12), 2012, pp. 22–28.

[3] J. Bo et al., "MobileTest: A Tool Supporting Automatic Black Box Test for Software on Smart Mobile Devices," Proc. 2nd Int'l Workshop on Automationof Software Test (AST 07), 2007, pp. 8–14.

[4] S. Anand et al., "Automated Concolic Testing of Smartphone Apps,"Proc. ACM SIGSOFT 20th Int'l Symp. Foundations of Software Eng. (FSE12), 2012, pp. 1–11.

[5] D. Amalfitano et al., "Using GUI Ripping for Automated Testing ofAndroid Applications," Proc. 27th IEEE/ACM Int'l Conf. Automated SoftwareEng. (ASE 12), 2012, pp. 258–261.

[6] T. Kallio and A. Kaikkonen, "Usability Testing of Mobile Applications:A Comparison between Laboratory and Field Testing," J. UsabilityStudies, vol. 1, no. 1, 2005, pp. 4–16.

[7] R. Mizouni et al., "Performance Evaluation of Mobile Web Services,"Proc. 9th IEEE European Conf. Web Services (ECOWS 11), 2011,pp. 184–191.

[8] T. Puhakka and M. Palola, "Towards Automating Testing of CommunicationalB3G Applications," Proc. 3rd Int'l Conf. Mobile Technology,Applications & Systems, 2006, article no. 27, pp. 1–6.

[9] I. Satoh, "Software Testing for Wireless Mobile Computing," IEEE WirelessComm., vol. 11 , no. 5, 2004, pp. 58–64.

[10] H. Song et al., "An Integrated Test Automation Framework for Testingon Heterogeneous Mobile Platforms," Proc. 1st ACIS Int'l Symp. Softwareand Network Eng., 2011, pp.141–145.

[11] E. Giordano et al., "MoViT: The Mobile Network Virtualized Testbed,"Proc. 9th ACM Int'l Workshop Vehicular Inter-networking, Systems, andApplications, 2012, pp. 3–12.

[12] C. Q. Tao, J. Gao, "Modeling Mobile Application Test Platform and Environment: Testing Criteria and Complexity Analysis", Proc. the 2014 Workshop on Joining AcadeMiA and Industry Contributions to Test Automation and Model-Based Testing, 2014, pp 28-33.

[13] R. Buyya, et al., "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities", Proceedings of International Conference on the High Performance Computing & Simulation (HPCS), 2009, pp 1-11.

[14] W.T. Tsai, Y. Hang, and Q. H. Shao, "Testing the Scalability of SaaS Applications", Proc. IEEE International Conference on Service-Oriented Computing and Applications (SOCA), 2011, pp 1-4.

[15] W. Hargassner, et al.,"A Script-Based Testbed for Mobile Software Frameworks", Proc. of International Conference on Software Testing, Verification, and Validation, 2008, pp 448-457.

[16] I. Satoh, "A Testing Framework for Mobile Computing Software". IEEE Transaction on Software Engineering, Vol. 29(12), 2012, pp 1112-1121.

[17] H. Muccini, A. D. Francesco, and P. Esposito, "Software Testing of Mobile Applications: Challenges and Future Research Directions", Proc. International Workshop on Automatic Software Test Automation, 2012, pp 29-35.

[18] J. Gao, et al., "Mobile Application Testing: A Tutorial", IEEE Computer, 47(2), 2013, pp 46-55.

[19] J. Hartikainen, (July, 2013) Why use user story based testing Retrieved (July, 2014) from http://codeutopia.net/blog/2013/07/28/why-use-user-story-based-testing-tools-like- cucumber-instead-of-other-tddbdd-tools/.

[20] C. Siemens, (November 2013) The Search of Mobile App Test Automation. Retrieved (June, 2014) from http://engineering.zillow.com/the-search-for-mobile-app-test-automation/.

[21] Testing Methods and Tools. Retrieved (July, 2014) from http://www.methodsandtools.com/tools/tools.php

[22] B. Phar, Hooking into the Test Process. Retrieved (July, 2014) from http://docs.behat.org/guides/3.hooks.html

[23] TDD Style of software development Retrieved(Aug, 2014) from www.ibm.com

Acknowledgement

This paper is supported by the National Natural Science Foundation of China under Grant No.61402229, 61502233, and 61602267; the Open Fund of the State Key Laboratory for Novel Software Technology (KFKT2015B10); the Postdoctoral Fund of Jiangsu Province under Grant No.1401043B, and the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant no. 15KJB52003.

DOI reference number: 10.18293/SEKE2017-170

Random GUI Testing of Android Application Using Behavioral Model

Woramet Muangsiri*, Shingo Takada*
*Dept. of Information and Computer Science, Keio University, Yokohama, Japan
{w.muangsiri, michigan}@doi.ics.keio.ac.jp

***Abstract*—Automated GUI testing based on behavioral model is one of the most efficient testing approaches. By mining user usage, test scenarios can be generated based on statistical models such as Markov chain. However, these works require static analysis before starting the exploration which requires too much prerequisites and time. In this work, we propose a behavioral-based GUI testing approach for mobile applications that achieves faster and higher coverage. Our approach does not conduct static analysis. It creates a behavioral model from usage logs by applying a statistical model. The events within the behavioral model is mapped to GUI components in a GUI tree. Finally, it updates the model dynamically to increase the probability of an event that rarely or never occurs when users use the application. We evaluated our approach on three open-source Android applications, and compared it with other approaches. Our approach showed the effectiveness of our tool.**

Keywords- Testing Tools; Testing Automation; GUI Testing; Android; Behavioral Model;

I. INTRODUCTION

It is undeniable that the smartphone has become a part of our life, and adoption has reached almost 3 billion active smartphone users in 2016 and still increasing year by year. Due to the huge and intensely competitive market, the high-quality and flawless graphic user interface is an essential part of success. Moreover, the mobile application design trend has been evolving rapidly. GUI testing is a very important part of achieving a high-quality application.

Various methods for automated GUI testing have already been studied, implemented, and evaluated [1]. However, according to a study conducted by Joorabchi et al [2], more than 70% of the survey respondents preferred to adopt manual GUI testing in practice, and less than 5% are engaged in fully automated GUI testing. Moreover, more than half of the participants admitted that GUI testing is challenging to automate and is still labor intensive.

To overcome these challenges, our goal is to create an automated tool that can be adopted easily, with smaller dependency and higher efficiency. We propose a behavioral-based automated GUI testing approach which requires as little setup effort as possible, while still achieving equivalent or better code coverage than other automated approaches by:

1) Avoiding static analysis by mapping the GUI tree and behavior model during runtime.
2) Updating the model dynamically to increase the probability of an event that rarely or never occurs when the user uses the application.
3) Applying a statistical model to create a behavioral model.

The remainder of this paper is organized as follows: Section 2 provides related works on GUI testing, Section 3 describes the methodology as well as the architecture, Section 4 evaluates our approach, Section 5 outlines the limitations of our tool, and Section 6 contains our conclusions and future works.

II. BACKGROUND AND RELATED WORKS

There are a number of works related to automated GUI testing for the Android platform. One of these is a tool called Monkey [3] which is bundled with Android SDK and is the most frequently used tool to test Android apps.

Android Monkey generates and sends pseudo-random streams of GUI events to the *Application Under Test* (AUT). It is fully automatic, and it can efficiently test the AUT with a large number of simple inputs. However, it is not suited for generating inputs that require human intelligence, and it tends to generate redundant inputs.

Other approaches such as model-based approaches [4], [5] have been proposed. Dynodroid [5] is based on random exploration with several strategies that make its exploration more efficient compared to a uniform distribution random testing. Frequency strategy selects the events that have been least frequently selected so far. Biased random strategy, like the frequency strategy, maintains a history of how often each event has been selected, but in a context-sensitive manner.

The model-based exploration strategies consider each individual activity as a state, and each event as a possible transition. Amalfitano et al. [4] presented *AndroidRipper*, an automated GUI-based tool to test Android apps in a structured manner. It implements a depth-first search (DFS) strategy and restarts the exploration from a fresh starting state when it cannot go to any other states during the exploration.

Recently, several techniques [6], [7], [8], [9] which captures user's input have been proposed. Linares-Vasquez, et al. [8] created a tool called *MonkeyLab* which mines GUI-based models that are used to generate actionable scenarios for both natural and unnatural sequences of events. Gomez et al. [9] proposed a crowdsourced approach and applied Path Analysis and Sequential Patterns algorithms to reproduce context-sensitive crashes by real users.

DOI reference number: 10.18293/SEKE2017-099

III. METHODOLOGY

Although the basis of our approach of using a behavioral model is not new, our approach is unique in that we continually update the behavioral model as the test progresses. The initial behavioral model is created from usage logs using statistical modeling, but we also consider events that did not appear in the usage logs. This is done through mapping events in the behavioral model with GUI components in the GUI tree. Event probabilities are adjusted as the test progresses.

Fig. 1 shows the overall architecture of the system. In this paper, the photo editing application in Fig. 2 will be used as a running example. Our approach can be divided into six steps as follows:

1) Gather the usage logs beforehand and extract GUI tree from the AUT (Section III-A).
2) Apply a statistical model to create a behavioral model, then map with GUI Tree (Section III-B).
3) Select an event from the behavioral model randomly (Section III-C).
4) Fire the selected event to the AUT (Section III-D).
5) Adjust the probability by frequency and update the model if needed (Section III-E).
6) Repeat from (2) to (5) until reaching a time limit or pre-decided number of events.

A. Observation

The purpose of this step is to acquire the mandatory components, the usage logs and GUI tree, for generating behavioral model.

1) Obtain Usage logs: This is the only prerequisite step before runtime. Usage logs are obtained from the AUT by using *Recorder*, the usage recording service. The *Recorder* service was implemented using Android's *AccessibilityService*[1] which captures events executed by the testers or users in the form of *AccessibilityEvent*. These events are then collected and transferred to the *Observer* through *Android Debug Bridge* (ADB). We focus on the events shown in Table I. Once the user starts using the application while the *Recorder* service is turned on, a recording session starts, which will end when the user closes the application. Since this data is the basis for the behavioral model, it is better to have many sessions. One way to obtain a large amount of sessions is to crowdsource [9]. This process does not require the developer to modify AUTs source code.

2) Extract GUI Tree from AUT: The GUI tree is a hierarchy of GUI components, extracted from the AUT during runtime using Android's *Hierarchy Viewer*[2]. We chose this dynamic approach, rather than statically analyzing the AUT's apk file; this results in adding less than $100ms$ overhead per action. Fig. 3 shows a simplified GUI tree from the example activity in Fig. 2. In this example, the simplified GUI Tree includes 6 GUI components, e_1 *backButton,* e_2 *editButton,* e_3 *deleteButton,* e_4 *shareButton,* e_5 *imageView, and* e_6 *saveButton*. In the GUI tree, each component contains its *type, resource-id, text, coordinate,* and *possible actions* that the user can take.

[1]https://developer.android.com/training/accessibility/service.html
[2]https://developer.android.com/studio/profile/hierarchy-viewer.html

TABLE I
ACCESSIBILITY EVENT TYPES

Event type	Description
TYPE_VIEW_CLICKED	Represents the event of clicking on a *View*.
TYPE_VIEW_LONG_CLICKED	Represents the event of long clicking on a *View*.
TYPE_VIEW_SCROLLED	Represents the event of scrolling on a *View*.
TYPE_VIEW_TEXT_CHANGED	Represents the event of changing the text of an *editText*.
TYPE_WINDOW_STATE_CHANGED	Represents the event of opening a *PopupWindow, Menu, Dialog, etc*.

B. Behavioral Model Generation and Mapping

In this step, the behavioral model is created from usage logs, then events are mapped to GUI components in the GUI tree.

1) Behavioral Model Generation: The behavioral model is created from usage logs by approximating the conditional probabilities using the *Markov assumption* (see a simplified behavioral model in Fig. 4). We use 5-grams model. An n-gram model is simply a sequence of words. In the context of software testing, these words are events. The *n* refers to the number of events. Consider a sequence of events $e_{tap1}, e_{tap2}, e_{hold1}, e_{swipeUp}$. If $n = 2$, then the 2-gram or *bigram* would be:

1) $p(e_{tap2}|e_{tap1})$
2) $p(e_{hold1}|e_{tap2})$
3) $p(e_{swipeUp}|e_{hold1})$

And for $n = 3$, the *3-grams* or *trigram* would be:

1) $p(e_{hold1}|[e_{tap1}, e_{tap2}])$
2) $p(e_{swipeUp}|[e_{tap2}, e_{hold1}])$

Where $p(e_x|[e_j, e_k])$ is the probability of event e_x being selected given history events $[e_j, e_k]$. With larger n, a model can store more context with a well-understood space-time tradeoff. It is possible to use higher than 5-gram if more sessions of usage log and processing power are available.

In practice, the app might not be entirely explored by users. In Fig. 4, suppose that there is another event e_6 that can occur when the user is at Activity B, but it is missing from the simplified behavioral model. This means that the model is not fully covered and that event e_6 will never be mapped, selected, and executed by our tool. We call the event that is available in the GUI tree but does not exist in the model *an unknown event*.

To overcome this problem, it is necessary to adjust the probability distributions. We use *KenLM* [10], which is a fast and low-memory language model toolkit. It includes the modified Kneser-Ney [11] technique, which is considered to be an effective smoothing technique. The three key ideas of the technique are:

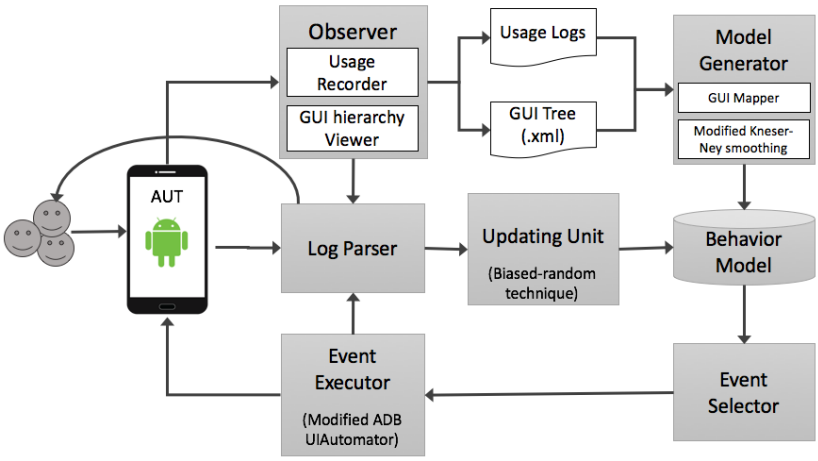


Fig. 1. An overview of the tool's architecture.

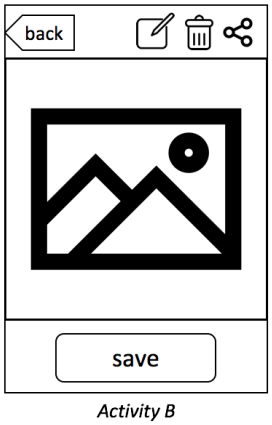


Fig. 2. The Example Application Under Test

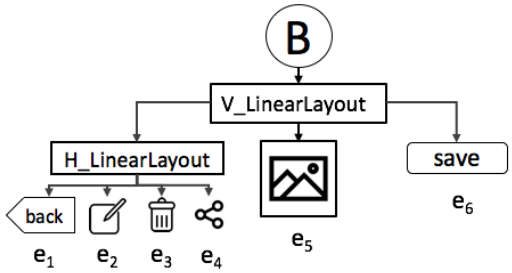


Fig. 3. A Simplified GUI Tree

- Absolute-discounting: in order to give an unknown event some probability, we must reduce the probability of the others in order to retain a valid probability distribution.
- Interpolation: recursively combine probabilities for all k-grams where $k \in 1, \ldots, n$. If an event sequence has any k-gram suffix appear in the model, it will yield a non-zero probability.
- Histories: the number of contexts that each event appears in should be taken into account. For instance, if an event occurs only in a specific context, it should be less likely to appear in a novel context.

Fig. 5 shows one example behavioral model after smoothing. The unknown event e_6, as well as rare events e_1 and e_5 receive some probability mass from the others.

2) Mapping behavioral model with GUI Tree: We do not actually merge an event and a GUI component together. Each event within the behavioral model is mapped to a GUI component within the GUI tree using a unique *resource-id* as a key, or with *type* and *text* if *resource-id* is not defined. The GUI mapper is implemented using a hash table for quicker lookup. The table is stored in main memory and updated each time after *Observation* (Section III-A). After selecting an event, the mapper looks up the table and returns the corresponding GUI fragment to the event.

C. Selection

The purpose of our tool is to test the application, not to create a natural action sequence. Therefore, the next event e_{next} will be randomly selected instead of naively picking the event with the highest probability. Let $e_1, e_2, e_3, \ldots, e_n \in \{possibleEvents\}$ and h be the history of previously selected events. e_{next} will be randomly selected by *Event Selector* based on the smoothed probability of an event given a sequence of previous selected events:

$$e_{next} = \omega([P(e_1|h), P(e_2|h), P(e_3|h), \ldots, P(e_n|h)])$$

where $\omega(L)$ is the weighted random function of a list L, $[P(e_1|h), P(e_2|h), P(e_3|h), \ldots, P(e_n|h)] \in L$, $\sum_{i=1}^{n} P(e_i|h) = 1$, and $P(e_x|h)$ is the probability of event e_x occurring given history h after updating (see Section III-E). From Fig 5, L should be *[0.07, 0.14, 0.28, 0.41, 0.07, 0.03]* for e_1 to e_6, respectively. We then calculate partial sum of the list to be *[0.07, 0.21, 0.49, 0.90, 0.97, 1.00]*. ω will uniformly random generate a number between 0.00 and 1.00 and select an e_x where the generated number is less than or equal to

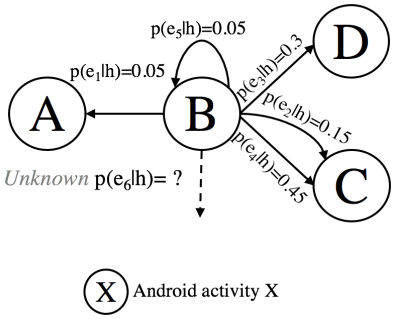


Fig. 4. A Simplified Behavioral Model

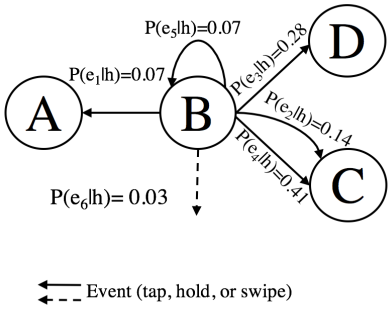


Fig. 5. Behavioral Model after smoothing

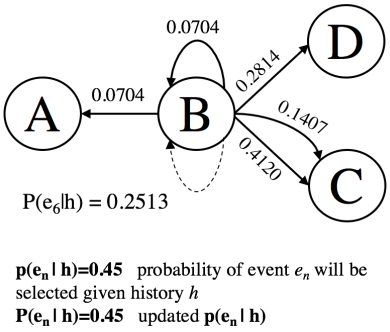


Fig. 6. Behavioral Model after updating

its partial sum but greater than e_{x-1}. Suppose the number is 0.99. In this case, e_{next} is e_6.

D. Execution

The *Event Executor* executes the e_{next} that was chosen by *Event Selector* on an actual mobile device or an emulator by *UIAutomator*, which is a custom wrapper around *Android Debug Bridge* written in python. Our tool can execute events such as Clicking, Long-Clicking, Scrolling, and Texting. In our tool, these events are defined as follows:

- Clicking: to tap and release immediately at the center of the view.
- Long-Clicking: to tap and hold 3 seconds before releasing at the center of the view.
- Scrolling: to tap and drag to $(x + \Delta x,\ y + \Delta y)$ position.
- Texting: to input an arbitrary pre-generated string[3]

Suppose e_{next} is the e_6 from Fig. 4, and it is clickable. The Event Executor will fire a *clicking event* to the center of the view, which in this case would be *saveButton*.

E. Updating

After execution, the model will be updated by using biased random technique to adjust selected/executed event's probabilities. It reduces a factor of small delta δ from the previously selected and executed event:

$$p\prime(e_x|h) = p(e_x|h) - d(e_x, \delta f_x) \tag{1}$$

where f_x is the frequency of e_x, $d(e_x, \delta f_x)$ is the discounted value of e_x, and $p\prime(e_x|h)$ is the probability of event e_x given history h. Suppose $d(e_6, \delta f_6) = 0.05$. Using equation (1), $p\prime(e_6|h)$ is 0.025.

After reducing some probability from an e_x, each $p\prime$ probability must be recalculated in order to retain a valid probability distribution by dividing it with total $p\prime$ probability.

$$P(e_x|h) = \frac{p\prime(e_x|h)}{\sum_{i=1}^{n} p\prime(e_i|h)} \tag{2}$$

The results of equation(2) are shown in Fig 6. The probability of $P(e_6|h)$ decreases. On the other hand, $P(e_1|h)$ to $P(e_5|h)$ slightly increases. This will allow other events to have a greater chance to be selected which should lead to higher code-coverage.

Finally, after executing e_6 and repeating the observation step (Section III-A), the *Updater* updates the behavioral model (Fig. 6).

[3]https://github.com/minimaxir/big-list-of-naughty-strings

TABLE II
LIST OF APPS USED IN OUR EVALUATION

App	Version	#LOC	Desc
Anymemo	8.3.1	8989	a flash card learning software.
World Clock	0.6	1251	a simple clock around the world app.
Weight Chart	1.0.4	1149	a log keeper application of body weight and display on a graphical chart.

IV. EVALUATION

We conducted a case study to evaluate our approach focusing on code coverage, and compared it to three popular approaches: Android Monkey, manual testing, and Dynodroid tool.

A. Target Application

We target three unmodified open-source Android applications: Anymemo[12], World Clock[13], and Weight Chart[14]. We chose these apps since they have dynamic content, static content, and complex user interface, respectively.

Table II shows the version numbers and short description of the applications.

B. Experiment setup

All our experiments were done using Android API 19, the current baseline version for application development, 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 i7 processors and 16 GB of RAM. The usage logs were collected by asking five computer science students to use each application for five minutes. No restrictions were placed on how they were to use the applications. For each experiment session, an AUT was installed on a freshly-created emulator

with only default setting. Each experiment session for our tool, Android Monkey, and Dynodroid consists of 5,000 touchable events and we added a delay between events of 500ms to ensure that screen transition or animation has completed before executing the next event. For manual testing, we asked an Android user to manually exercise these apps as much as possible within 40 minutes. After each experiment session, the emulator was destroyed to prevent it from affecting later sessions. We used Emma [15] and custom shell script to collect coverage measurement from AUT.

C. Results and Discussion

The results of our study are summarized in Table III. We collected code coverage at 500 event intervals for a total of 10 data collection points. The most left column denotes the testing approaches including our approach, Android Monkey, Manual testing by human, and Dynodroid, respectively. The numbers in the table show the number of lines and percentage of code coverage by total lines of code (see Table II) that were covered by each approach for the three applications tested. Our tool outperformed Android Monkey and Dynodroid in *Anymemo* but did not outperform manual testing.

Manual testing easily outperformed our tool and other approaches, achieving the highest coverage for all three applications. This can be expectable, given that the human can provide more intelligent text or sequential inputs. Fig. 7 shows the accumulated code coverage for *Anymemo*. The X axis denotes the number of events based on the 500 event intervals. The plots for manual testing are the final result since we did not record coverage during manual testing. This is because we did not want to disturb our human subjects during the session. Due to our behavioral model, our tool successfully completed a series of operations (in this specific case, reviewing flash cards), and was able to conduct further operations; but Android Monkey and Dynodroid could not attain this level of operations within 5000 events. However, our tool failed to generate a valid string for sensitive cases, for instance, import and export path, since our tool records only the event type, and not a particular string.

The results for *World Clock* and *Weight Chart* are just slightly higher than Android Monkey. Moreover, in *Weight Chart*, our tool significantly underperformed manual testing and slightly lower than Dynodroid. A possible issue might be the fact that our tool cannot detect and perform complex gesture such as pinching or dragging to the Canvas components [16]. For example, in *Weight Chart*, the possible actions for a *display graph*, a line graph for showing weight data, were not detected by our tool. This resulted in several actions to not be reached.

D. Threats to Validity

There are potential threats to validity of our results. The first threat to validity would be the use of students as the human subjects. Since the task was to compare the coverage from manual testing and other approaches, there is a possibility that professional testers are better at exercising AUT.

TABLE III
ACCUMULATED CODE COVERAGE AFTER 5000 EVENTS OR 40 MINUTES. THE STATS INCLUDE THE CODE COVERAGE FROM OUR TOOL (#COV), FROM ANDROID MONKEY (#MCOV), FROM MANUAL TESTING BY HUMAN (#HCOV), AND FROM DYNODROID (#DCOV)

App	Anymemo	World Clock	Weight Chart
#COV	2885 (32.1%)	1120 (89.5%)	608 (52.9%)
#MCOV	1856 (20.6%)	1023 (81.9%)	536 (46.6%)
#HCOV	4173 (46.4%)	1149 (91.8%)	986 (85.8%)
#DCOV	1327 (14.8%)	1096 (87.6%)	673 (58.6%)

The second threat to validity is the three applications we used. We chose the three because they were open-source, have several categories, and were used by various studies[5], [1]. However, the sizes of our subject applications are relatively small compared with the top applications in Google Play[4]. Further evaluation with larger applications should be conducted.

The third threat to validity is the usage logs for a behavioral model. We believe that testing the app casually vs seriously gives a different result. Since the benefit is not clear, we did not provide particular restrictions or goals during the recording sessions.

V. LIMITATIONS OF OUR TOOL

This section outlines the current limitations of our tool.

A. Minimum Android API level

Our tool works on Android API level 16(Version 4.1.x) and onward. However, this is not significant since only 2% of Android device are API level 15 and below[17].

B. Scrolled Events need to be throttled

Since the scrolled events are emitted from the Android's *AccessibilityService* constantly while the user is scrolling, a scroll event usually becomes repetitive events which disrupt the model's probability distribution. To prevent that from happening, our tool merges consecutive scroll events into a single scroll event. However, there is a chance that two consecutive but different scrolls are merged unintentionally.

C. Resource-id must be provided

It is possible that the GUI component's *resource-id* is not defined by developers. In this case, other attributes are used for mapping (see Section III-B2).

D. Canvas component's properties are not detected

We rely on *UIAutomator* to dump the GUI hierarchy during runtime. It fails to extract a canvas component's properties, which prevents our tool from executing corresponding GUI events.

[4]https://play.google.com

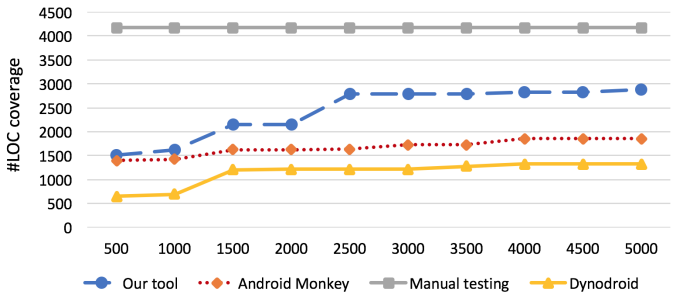


Fig. 7. Accumulated code coverage for Anymemo

VI. Conclusion and Future Work

In this paper, we presented our on-going work on automating GUI testing for Android applications. Our approach combines manual testing aspects by creating a behavioral model based on actual usage logs. We map the behavioral model with GUI tree in runtime, and handle the zero probability *unknown events* with *modified Kneser-Ney smoothing* technique.

We have implemented and conducted a comprehensive evaluation of our approach. We compared it with existing Android testing approach such as Android Monkey, manual testing, Dynodroid on open-source Android applications. Our approach outperforms Android monkey for all applications, and state-of-the-art tool Dynodroid for two out of three applications, but it is still behind manual testing.

For the future works, we will extend the actionable events that our tool can handle such as Enabling, Pinching, and Dragging. We also plan on extending our tool based on the limitations given in section V. We intend to evaluate our tool with larger Android application in the Google Play. Finally, we will further investigate our tool in term of time to reach the saturation point [18], and ability to find bugs.

Acknowledgement

This work was supported by JSPS KAKENHI JP15K00104.

References

[1] S. R. Choudhary, A. Gorla, and A. Orso, "Automated test input generation for android: Are we there yet?" in *Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE Computer Society, 2015, pp. 429–440.

[2] M. E. Joorabchi, A. Mesbah, and P. Kruchten, "Real challenges in mobile app development," in *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, 2013, pp. 15–24.

[3] UI/Application Exerciser Monkey — Android Developers. [Online]. Available: http://developer.android.com/tools/help/monkey.html

[4] D. Amalfitano, A. R. Fasolino, P. Tramontana, S. De Carmine, and A. M. Memon, "Using gui ripping for automated testing of android applications," in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2012, pp. 258–261.

[5] A. Machiry, R. Tahiliani, and M. Naik, "Dynodroid: an input generation system for android apps," *9th Joint Meeting on Foundations of Software Engineering*, p. 224, 2013.

[6] P. A. Brooks and A. M. Memon, "Automated gui testing guided by usage profiles," in *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*. New York, NY, USA: ACM, 2007, pp. 333–342.

[7] J. L. San Miguel and S. Takada, "Gui and usage model-based test case generation for android applications with change analysis," in *Proceedings of the 1st International Workshop on Mobile Development*. ACM, 2016, pp. 43–44.

[8] M. Linares-Vásquez, M. White, C. Bernal-Cárdenas, K. Moran, and D. Poshyvanyk, "Mining android app usages for generating actionable gui-based execution scenarios," in *Proceedings of the 12th Working Conference on Mining Software Repositories*, ser. MSR '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 111–122.

[9] M. Gómez, R. Rouvoy, B. Adams, and L. Seinturier, "Reproducing context-sensitive crashes of mobile apps using crowdsourced monitoring," in *Proceedings of the International Conference on Mobile Software Engineering and Systems*. ACM, 2016, pp. 88–99.

[10] K. Heafield. KenLM Language Model Toolkit. [Online]. Available: http://kheafield.com/code/kenlm/

[11] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," *Computer Speech Language*, vol. 13, no. 4, pp. 359 – 394, 1999.

[12] H. Ning. AnyMemo. [Online]. Available: https://anymemo.org

[13] R. Agarwal. World Clock Android app. [Online]. Available: https://github.com/rahulaga/WorldClock

[14] Senselesssolutions. Weight Chart. [Online]. Available: https://github.com/bluezoot/weight-chart

[15] EMMA: a free Java code coverage tool. [Online]. Available: http://emma.sourceforge.net

[16] Canvas. [Online]. Available: https://developer.android.com/reference/android/graphics/Canvas.html

[17] Dashboards — Android Developers. [Online]. Available: https://developer.android.com/about/dashboards/index.html

[18] D. Amalfitano, N. Amatucci, A. R. Fasolino, P. Tramontana, E. Kowalczyk, and A. M. Memon, "Exploiting the saturation effect in automatic random testing of android applications," in *Proceedings of the Second ACM International Conference on Mobile Software Engineering and Systems*, ser. MOBILESoft '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 33–43.

Localization of Linearizability Faults on the Coarse-grained Level

Zhenya Zhang, Peng Wu, Yu Zhang
State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
University of Chinese Academy of Sciences

***Abstract*—Linearizability is an important correctness criterion that guarantees the safety of concurrent data structures. Due to the nondeterminism of concurrent executions, reproduction and localization of a linearizability fault still remain challenging. The existing work mainly focuses on model checking the thread schedule space of a concurrent program on a fine-grained (state) level, and hence suffers from the severe problem of state space explosion. This paper presents a tool called CGVT to build a small test case that is sufficient enough for reproducing a linearizability fault. Given a possibly long history that has been detected non-linearizable, CGVT first locates the operations causing a linearizability violation, and then synthesizes a short test case for further investigation. Moreover, we present several optimization techniques to improve the effectiveness and efficiency of CGVT. We have applied CGVT to 10 concurrent objects, while the linearizability of some of the concurrent objects is unknown yet. The experiments show that CGVT is powerful and efficient enough to build the test cases adaptable for a fine-grained analysis.**

I. Introduction

Linearizability [1] is a widely accepted correctness criterion that guarantees the safety of concurrent data structures or concurrent objects. Intuitively, a concurrent history of a shared object is linearizable if each operation of the object appears to take effect instantaneously at some point, called linearization point, between the invocation and response of the operation, and the history can be serialized as a sequence of the operations that is consistent with the sequential specification of the object. Linearizability checking is still a challenging task for concurrent objects.

Due to the nondeterminism of concurrent executions, reproduction and localization of a linearizability fault is notoriously difficult. This problem can be addressed by systematically exploiting all possible fine-grained traces that are composed of memory access instructions [2], [3], which however suffers from the severe problem of state space explosion. Although techniques such as iterative context bounding [4] can help improve the scalability of this systematic testing approach, it would work better with small test cases. Besides, such test cases can be generated through static analysis or empirical evidence [5], [6], but lack accuracy, especially for very complicated objects or concurrency faults. ConTeGe [7] can fully automatically produce a large number of small-scale traces, which are composed of operations with random arguments, but may not manifest potential faults. VeriTrace [8] can produce traces that can trigger various buggy executions, but these traces may contain redundant operations, therefore raise the complexity of further analysis unnecessarily.

In this paper, we present a tool, CGVT, to build small test cases that are sufficient enough for triggering linearizability faults. Given a possibly long history that has been detected non-linearizable, CGVT first locates the operations causing a linearizability violation. Note that the operations resulting in a non-linearizable execution are usually not located as would be expected intuitively.

Example 1 Fig. 1 shows a simplified PairSnapShot [9], where an array d (of size 2) is shared between two concurrent threads. A `write(i,v)` operation writes `v` to `d[i]`, while a `read` operation reads the values of `d[0]` and `d[1]` together as a pair. A correctness criterion is that `read` should always return the values of the same moment. However, Fig. 2 illustrates a concurrent execution in which the return values of `read` do not exist at any moment of the execution. The 'x' labels in Fig. 2 mark the moments when each instruction takes effect. Then, this concurrent execution with the 5 operations explains exactly the cause of the linearizability violation on a fine-grained state level.

```
PairSnapShot:

  int d[2];

  write(i,v){
    d[i] = v;   #1
  }
```

```
Pair read(){
  while(true){
    int x = d[0];    #2
    int y = d[1];    #3
    if(x == d[0])    #4
      return (x,y);
  }
}
```

Fig. 1: Source code of PairSnapShot

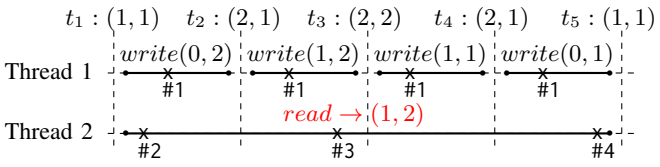


Fig. 2: A buggy trace of PairSnapShot

Example 2 Fig. 3 presents a linked-list based set [10]. Fig. 4 shows a non-linearizable trace of the set, where O_1 and O_2 appear to add 8 and 9, respectively, to the list, but later O_3 does not find 8 in the list. Herein, the synchronized block #3 can be treated as an atomic instruction. The actual cause

DOI reference number: 10.18293/SEKE2017-145

of the linearizability violation is as follows: after both add operations locate the same nodes pred and curr, add(8) first sets pred.next to node 8, then add(9) sets it to node 9, which makes node 8 removed from the list. It can be seen that although it is the wrong return value of contain(8) that reveals the linearizability violation, the root cause of this non-linearizable trace only lies in the concurrent add operations.

```
Set:
    add(int key){
        Node pred,curr = locate(key);   #1
        if(curr.key == key)             #2
            return false;
        synchronized(){                 #3
            Node n = new Node(key);
            n.next = curr;
            pred.next = n;
            return true;
        }
    }
```

Fig. 3: Source code of a buggy set

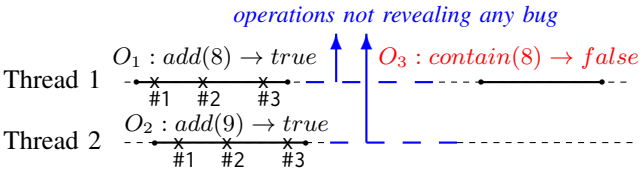


Fig. 4: A buggy concurrent trace of Set

Example 1 shows a scenario where a violation is "revealed" immediately after the corresponding linearizability fault is "triggered", while Example 2 shows another scenario where a linearizability fault is "triggered" possibly long before the corresponding violation is "revealed". In this work, we aim to localize, from a given non-linearizable trace, the operations that essentially cause the linearizability violation and then synthesize a small test case that can trigger the same linearizability fault.

Contributions. The main contributions of this paper are threefold.

- We make clear the relationship between "triggering" and "revealing" linearizability faults, and formally define *minimum test cases* that are sufficient to trigger linearizability faults;
- We propose the framework of CGVT for building minimum test cases, as well as a D-PCT based optimization and a heuristic rule based acceleration technique;
- We implement a prototype tool, CGVT, which was experimented with 10 concurrent objects. The experiment results show that the tool is effective and efficient enough to build the test cases adaptable for a fine-grained analysis.

Related work. The existing work on linearizability checking mainly aims to solve the state space explosion problem to accelerate the checking of concurrent histories. Automated linearizability checking algorithms [11], [12] suffer from a performance bottleneck. Based on [12], optimized algorithms were proposed through partial order reduction [13] or compositional reasoning [6]. Model checking was applied for linearizability checking, with simplified first-order formulas that can help improve efficiency [5], [14]. Fine-grained traces were introduced in [8] to accelerate linearizability checking.

The output of our work is a small test case that is adaptable for concurrency fault analysis on a fine-grained level, such as [2]–[4]. A framework with interleaving test generation heuristics was introduced in [2] for bug detection and replay. A constraint-based symbolic analysis method was proposed in [3] to diagnose concurrency bugs. An acceleration technique was presented in [4] for model checking thread schedule spaces. Unlike these bug reproduction and localization techniques [2], [4], [7] that aim at general concurrency bugs, our work aims at linearzability violation specifically.

Our work can also facilitate debugging concurrency bugs. Recent work on the debugging of concurrency bugs often applies the techniques for sequential programs to concurrent programs, such as assertions [15], or breakpoints [16].

Organization. The rest of the paper is organized as follows. Section II introduces the trace model and the formal definition of minimum test cases. Section III presents the implementation of CGVT, with the D-PCT based optimization and the heuristic rule based acceleration technique. Section IV discusses the results of experiments. Section V concludes the paper.

II. Trace Model

A. Linearizability

Let $\mathbb{M}, \mathbb{T}, \mathbb{O}, \mathbb{V}$ respectively denote the set of operation names, thread identifiers, operation identifiers and values. Then, $C = \{m(v_a)^t_o : m \in \mathbb{M}, v_a \in \mathbb{V}, o \in \mathbb{O}, t \in \mathbb{T}\}$ and $R = \{m^t_o \rightarrow v_r : m \in \mathbb{M}, v_r \in \mathbb{V}, o \in \mathbb{O}, t \in \mathbb{T}\}$ represent the set of operation *invocation* events and *response* events, respectively. We denote the operation identifier of an event $e \in (C \cup R)$ as $\mathtt{op}(e)$. Events $c \in C$ and $r \in R$ match each other, denoted $c \diamond r$, if $\mathtt{op}(c) = \mathtt{op}(r)$. A pair of matching events forms an operation O, written as $m(v_a) \rightarrow v_r$.

A sequence $S = e_1 e_2 \cdots e_n \in (C \cup R)^*$ is *well-formed* if:

- Each response is preceded by a matching invocation: $e_j \in R$ implies $e_i \diamond e_j$ for some $1 \leq i < j \leq n$
- Each operation identifier is used in at most one invocation/response: $\mathtt{op}(e_i) = \mathtt{op}(e_j)$ and $1 \leq i < j \leq n$ implies $e_i \diamond e_j$

A well-formed sequence S can be treated as a partial order set $(H, \prec_H)$, where H is called a history (or trace) composed of the operations formed by matching events in S, and $\prec_H$ is the *happen-before* relation in S. For operations $O_1, O_2 \in H$, $O_1 \prec_H O_2$ if the response of O_1 is ahead of the invocation of O_2 in S. Let $\mathtt{size}(H)$ denote the *size* of H, i.e., the number of the operations involved in history H.

An invocation event is *pending* in H if no matching response event follows it. An *extension* of H, denoted $\mathcal{E}(H)$, is a history constructed by appending response events to all the pending invocation events in H. Sometimes we ignore these

pending invocation events and get $\mathcal{C}(H)$, the subsequence of H consisting of all the matching invocation and response events in H.

If H is a total order set, then H is *sequential*. A *specification* of an object is the set of all the sequential histories that satisfy the correctness criterion of the object.

Definition 1 (Linearizability)**.** *A history H of a concurrent object is linearizable if there is an extension $\mathcal{E}(H)$ and a history S in the specification of the object such that:*

- *Elements of $\mathcal{E}(H)$ and S are same;*
- *$\prec_H \subseteq \prec_S$, i.e., if $O_1 \prec_H O_2$, then $O_1 \prec_S O_2$.*

Here, S is called a witness of H.

B. Minimum test case

Operations with the same thread identifier form a sequential program P_s, an execution of which results in a sequential history. A concurrent program P_c with n threads is a set of n sequential programs. An execution of P_c is a concurrent execution of the n sequential programs, each starting at an arbitrary moment.

A *state* of an object at some moment is a mapping from the shared variables to their values. Considering the initial state of an object as default, we can use a sequential program P_s to represent a state, since P_s can determine it definitely. For concurrent program P_c, thread schedules decide what states are reached. A *schedule* for P_c is a sequence of memory access instructions in P_c.

A *test case* is defined as a triple $\mathtt{t_c} = (P_s, P_c, P_r)$, where P_s is a state, P_c is a concurrent program and P_r is a sequential program. Starting from state P_s, an execution of $\mathtt{t_c}$ runs the sequential programs in P_c concurrently to "trigger" a bug if any, and then runs P_r at last to "reveal" the bug. The trace of an execution of $\mathtt{t_c}$ depends on the schedule of the execution for P_c. If there exists a schedule leading a trace of $\mathtt{t_c}$ non-linearizable, we say that $\mathtt{t_c}$ is *potential to trigger a linearizability fault* or *buggy*; otherwise $\mathtt{t_c}$ is *correct*.

Definition 2 (minimum test case)**.** *A test case $\mathtt{t_c}$ is an minimum test case if it is potential to trigger a linearizability fault and removal of any operation in its P_c results in a correct test case.*

Examples The trace shown in Fig. 2 results from a minimum test case where $P_s = \{write(0,1), write(1,1)\}$, $P_c = \{\{write(0,2), write(1,2), write(1,1), write(0,1)\}, \{read\}\}$, and $P_r = \emptyset$. The trace shown in Fig. 4 results from a minimum test case where $P_s = \{add(7), add(10)\}$, $P_c = \{\{O_1\}, \{O_2\}\}$, and P_r is a sequential program ending with O_3.

III. CGVT

In this section, we present the basic implementation of CGVT, a D-PCT based optimization and a heuristic rule based acceleration technique. CGVT is divided into two phases, detection and localization. The former is to acquire a possibly long trace that is non-linearizable, and the latter is to locate the operations causing a linearizability fault, and synthesize a minimum test case for it.

A. Basic implementation

Detection phase. VeriTrace [8] is an off-line tool for automated linearizability checking. Given a concurrent object as an input, VeriTrace automatically runs concurrently a couple of threads, each executing a certain number of operations, and records the history H of the concurrent execution on the object. Then, WGL [13], a linearizability checking algorithm, enumerates possible sequential histories to search for a witness of the concurrent history H. If no witness of H is found, then H is detected non-linearizable and will be analyzed in the next phase for localization.

Localization phase. We define a *prefix* of the concurrent history H, denoted $H_p(n)$, as a sub-history of H composed of the former n events in H.

Theorem 1. *H is linearizable if and only if every prefix of H is linearizable.*

Proof For the "only if" direction, suppose that H is linearizable and there exists a non-linearizable prefix $H_p(k)$. Then, any extension $\mathcal{E}(H_p(k))$ is non-linearizable, which implies that the addition of operations $O' \in \mathcal{E}(H_p(k)) \setminus \mathcal{C}(H_p(k))$ can not serialize it. Neither can the addition of operations O'' such that $\forall m(m \in \mathcal{C}(H_p(k)) \wedge m \prec_H O'')$, because such O'' can only be ordered after the operations of $\mathcal{C}(H_p(k))$ in a witness. Therefore, there is no witness for any superset of $\mathcal{C}(H_p(k))$, which implies H is non-linearizable, a contradiction.

For the "if" direction, if every prefix of H is linearizable, it is obvious that H is linearizable. □

Algorithm 1 Bug Localization

```
1:  OpOrder. SortBySize()
2:  s_o ← |OpOrder|
3:  function Localize
4:      for i ← {s_o, ··· , 0} do
5:          P_s ← OpOrder[i]
6:          P_c ← H_p(b) \ P_s
7:          t_c ← New TestCase(P_s, P_c, ∅)
8:          if !Check(t_c) then break
9:          end if
10:     end for
11:     for j ← {2, ··· , size(t_c.P_c) − 2} do
12:         P_c ← t_c.P_c[0 : j]
13:         P_r ← OpOrder[s_o] \ t_c.P_s \ P_c
14:         t_c ← New TestCase(t_c.P_s, P_c, P_r)
15:         if !Check(t_c) then return t_c
16:         end if
17:     end for
18: end function
19:
20: function Check(t_c)
21:     while k ← {0, ··· , σ} do
22:         if !L_n(t_c.Execute()) then return false
23:         end if
24:     end while
25:     return true
26: end function
```

From Theorem 1, it can be seen that for the given non-linearizable history H, there exists a non-linearizable prefix $H_p(b)$ whose size is smaller than any other non-linearizable prefixes of H. It is obvious that $H_p(b)$ involves the operations triggering the linearizability fault, since otherwise $H_p(b)$ would be linearizable. We can use $H_p(b)$ to construct a minimum test case for a fine-grained bug localization algorithm.

Algorithm 1 presents the algorithm for the coarse-grained bug localization. $OpOrder$ is a set of witnesses of all prefixes of $H_p(b)$. We sort the witnesses in $OpOrder$ by their sizes (line 1). The process of building a minimum test case contains two steps:

1) Select a prefix of $H_p(b)$, serialize it as P_s (lines 4-5), and then set the remaining suffix of $H_p(b)$ to be P_c (line 6), check the test cases iteratively until a buggy one is found (lines 8-9);
2) Set a new P_c to be a prefix of the previous P_c (line 12), then serialize the remaining operations as P_r (line 13), and check the test cases iteratively until a buggy one is found (lines 15-16).

Note that the serialization in lines 5 and 13 is directly based on the sequential traces in $OpOrder$, and in practice, we can get both $H_p(b)$ and $OpOrder$ easily through the "cache" mechanism of the WGL algorithm in the detection phase.

Function CHECK (lines 20-26) is used for checking whether a test case is buggy, where L_n is a predicate judging the linearizability of a trace. Here, σ is a threshold for the number of times the test case is executed. Our experiments show that using the setting of σ in the detection phase is always enough to detect the fault in the localization phase.

B. D-PCT based implementation

D-PCT based detection. The concurrent executions induced by VeriTrace closely depend on the run-time environment. Theoretically, for a concurrent program that contains n threads, each executing at most k instructions, the total number of its schedules is $\frac{(nk)!}{(k!)^n} \geq (n!)^k$, exponentially dependent on n and k. It is difficult to detect the faults that only appear under specific (complicated) schedules.

In order to address this problem, we adapt PCT [17], a scheduling algorithm with a much higher probabilistic guarantee of finding bugs, as D-PCT (Dynamic PCT) and apply it to CGVT, as shown in Fig. 5.

- In Fig. 5(a), during a concurrent execution in the detection phase, THREADSWITCH requests are called to ask whether a thread switch should take place currently;
- In Fig. 5(b), D-PCT responds immediately by invoking the PCT algorithm with the number of instructions (k).

Originally in PCT, k is required no less than the number of the instructions so that PCT can generate a schedule covering all the instructions, while in CGVT, k can not be pre-specified before the execution. Here, we model the instructions as a *structure tree*, by which we calculate k dynamically, as shown in Fig. 5(c).

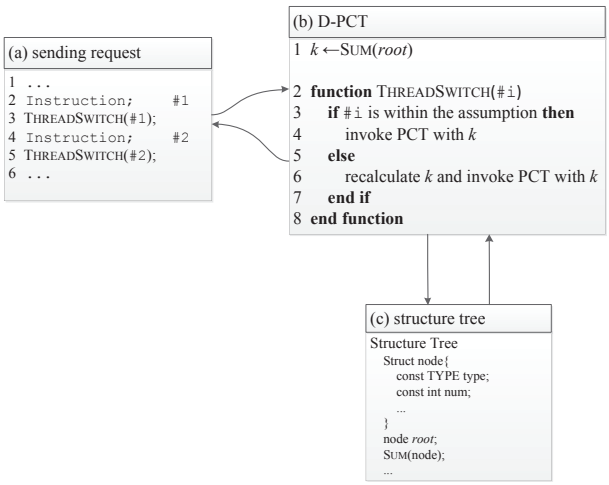


Fig. 5: Application of D-PCT

We split the instructions of an operation into different units: *sequential units* contain instructions executed sequentially such as `#1` in Fig. 1; *loop units* contain instructions executed iteratively such as `#2,#3,#4` in Fig. 1; *selection units* contain several *branches*, each containing instructions executed conditionally such as `#3` in Fig. 3. Moreover, each *loop unit* and each *branch* can further be split until all the instructions belong to a *sequential unit*. The instructions in an operation can be built as the following *structure tree*:

1) The root represents all the instructions of the operation;
2) Each child node of the root represents one of the three units, from left to right following the program order of the instructions;
3) Each child node of a *selection unit* node is one of its *branches*;
4) The child nodes of a *loop unit* or *branch* node represents its lower-level units until *sequential units* as leaves.
5) Each node holds two constants: `type` and `num`. `type` is one of `R, L, C, B, S` respectively representing root, *loop unit, selection unit, branch, sequential unit*. For a node of type `R, L, C` or `B`, `num` is the number of its child nodes; while for a node of type `S`, `num` is the number of the instructions in the *sequential unit*.

Example Fig. 6 presents the structure trees of the three operations in Fig. 1 and Fig. 3.

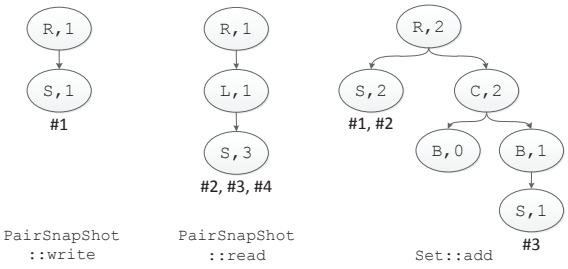


Fig. 6: Structure tree

Algorithm 2 Calculation of k

```
1: function SUM(node)
2:     value ← 0
3:     switch node.type
4:       case (S):                                  ▷ sequential unit
5:         value ← value + node.num
6:       break
7:       case (R|L|B):                  ▷ root, loop unit, branch
8:         for n ∈ node.child do
9:           value ← value + θ*SUM(n)
10:        end for
11:      break
12:      case (C):                                  ▷ selection unit
13:        for n ∈ node.child do
14:          value ← value + max(SUM(n) )
15:        end for
16:      break
17:    end switch
18:    return value
19: end function
```

With the help of a structure tree, D-PCT can make decision on thread switching as shown in Fig. 5(b). At first, D-PCT calculates k tentatively through function SUM (line 1), assuming that loop units iterate θ times and selection units select the branch with the maximum number of instructions. Then, each time a THREADSWITCH request is called, D-PCT checks whether the instruction, after which the request is called, is beyond the assumption (line 3). If not, D-PCT responds by invoking PCT directly (line 4); otherwise, D-PCT recalculates the currently remaining k in a way similar to line 1 and then invokes PCT (line 6).

Algorithm 2 presents the concrete algorithm of function SUM. It recursively searches the structure tree and adds up the number of the instructions of different units assuming that loop units iterate θ times (line 9) and selection units select the branch with the maximum number of instructions (line 14). Here, for a node of type L, if its number of iterations is known, θ is set accordingly, otherwise $\theta = 1$; for a node of type R or B, $\theta = 1$.

D-PCT based localization. In addition to Fig. 5(b), D-PCT records the instruction attached within each request to form the actual schedules during the detection phase. Then in the localization phase, these schedules are applied to function CHECK() in Algorithm 1 to guide the executions of P_c. In this way, the faults triggered under these schedules will be reproduced in the localization phase.

C. A heuristic rule based acceleration

We present a heuristic rule to make CGVT perform better. The rule is based on an **observation** that in P_c of a minimum test case (P_s, P_c, P_r), there exists a thread in which only one operation is executed.

This observation comes from the minimum test cases we have collected, which are presented in Table I. In Column 2, we call a test case is of type 1 if the test case "reveals" a fault immediately after the fault is"triggered", like the test case for PairSnapShot shown in Fig. 2; while a test case is of type 2 if the test case "reveals" a fault possibly long after the fault is "triggered", like the test case for the linked-list based set shown in Fig. 4. Column 3 reports the number of the operations needed in the concurrent threads of P_c in each minimum test case. Hence, we conjecture that whichever type a test case belongs to, one operation in some thread may be sufficient to trigger a fault. This observation is useful for acceleration if test cases built accordingly can still trigger faults.

TABLE I: Operation quantity on threads

	Type	OP Quantity
PairSnapShot [9]	1	1, 4
SimpleList [10]	2	1, 1
SimpleList (Size) [8]	1	1, 2
LockFreeList [18]	1	1, 1
K-Stack [19]	2	1, 1
OptimisticQueue [20]	1	1, 1
Snark [21]	1	1, 2
TreiberStack [5]	2	1, 3

IV. EXPERIMENT AND EVALUATION

Table II shows the experiment on 10 concurrent objects. Some of the objects are from the previous work and known to us, while others whose names begin with "BU" are adapted versions of the existing objects and unknown to us. The specification for the first 6 objects is *Set*. The specification for "OPQueue" and "BUQueue" is *FIFO-Queue*. The specification for the last two objects is *Stack*.

We compare the 3 versions of CGVT, respectively the basic version (B) in Section III-A, the D-PCT based version (D) in Section III-B, and the heuristic rule based version (H) in Section III-C. In the detection phase, for each object, n_s (possibly long) histories (Col. 4-6) are examined, each composed of $n_t * n_o$ (Col. 3) operations with n_t the number of the threads and n_o the number of the operations executed by each thread. Among these histories, n_l histories are detected non-linearizable (Col. 4-6). Note that the occurrence of a non-linearizable history is the prerequisite for localization, and the proportion reported in Col. 4-6 reflects the difficulty in detecting the linearizability faults. The average time costs for linearizability checking of these histories are listed in Col. 7-9. In the localization phase, selecting a non-linearizable history from the detection phase, CGVT works out minimum test cases of sizes shown in Col. 10-12. The time costs of localization are listed in Col. 13-15, mainly for the two calls of CHECK. The tool and benchmarks are available at *https://github.com/choshina/CGVT*.

We have the following observations from Table II:

1) The concurrent objects that interest us most are BUList3 and CGListS. For these two objects, version B cannot detect the bugs, but version D can. This shows that D-PCT enhances the CGVT's ability in detecting certain complicated bugs.
2) Comparing the time cost of version H with that of version D, we conclude that version H improves the efficiency.

TABLE II: Evaluation of CGVT

Object	Operations	Hist. size ($n_t * n_o$)	Detection phase						Localization phase					
			!L_n/Total (n_l/n_s)			Aver. time (ms)			$\mathtt{t_c}.P_c$ size			Time cost (ms)		
			B	D	H	B	D	H	B	D	H	B	D	H
LFList	add;remove	2*500	5/20	2/20	8/20	643	611	237	2	2	2	154	233	207
CGList	add;contain	2*50	20/20	20/20	20/20	33	24	23	2	2	2	51	256	219
CGList	add;remove;size	2*500	0/80	3/80	1/80	660	682	483	-	4	3	-	756	832
BUList1	add;remove	2*500	7/20	5/20	5/20	488	704	532	2	2	2	162	245	311
BUList2	add;remove	2*50	18/20	20/20	20/20	31	35	28	2	2	2	83	365	247
BUList3	add;remove	2*500	0/80	2/80	3/80	625	603	524	-	2	2	-	326	335
OPQueue	enqueue;dequeue	2*500	4/20	19/20	16/20	2430	1412	832	2	3	2	378	796	683
BUQueue	enqueue;dequeue	2*200	0/80	0/80	0/80	288	726	230	-	-	-	-	-	-
KStack	push;pop	2*50	10/20	18/20	15/20	83	30	62	2	3	3	2323	864	831
TBStack	push;pop	2*100	19/20	20/20	20/20	72	124	60	2	2	2	1596	532	356

Compared to version B, versions D and H perform better on the faults "revealed" long after "triggered" (K-Stack, TBStack).

3) From the results in the localization phase, it can be seen that CGVT delivers the test cases with no more than 4 concurrent operations. Such test cases are quite small-scale and adaptable for a fine-grained analysis.

V. Conclusion

This paper presents a tool, CGVT, for building a small test case that is sufficient to trigger a linearizability fault. CGVT firstly acquires a possibly large-scale concurrent history which has been detected non-linearizable, then locates the concurrent operations causing the linearizability violation, and synthesizes a minimum test case. A D-PCT based technique enhances the CGVT's ability in detecting complicated bugs, and a heuristic rule is introduced for accelerating detection and localization.

As future work, firstly, a tool for fine-grained localization is needed. Given a small test case, this tool ought to enumerate the fine-grained traces and investigate the data races on the shared variables to provide some guidance on bug repair. Secondly, we will try to apply CGVT and this tool to more concurrent programs, and extend their localization abilities from linearizability violation to other concurrency bugs.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China under Grants No.61100069, No.61161130530, and by the National Key Basic Research Program of China under Grant No.2014CB340701.

References

[1] M. P. Herlihy and J. M. Wing, "Linearizability: A correctness condition for concurrent objects," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 12, no. 3, pp. 463–492, 1990.

[2] O. Edelstein, E. Farchi, E. Goldin, Y. Nir, G. Ratsaby, and S. Ur, "Framework for testing multi-threaded java programs," *Concurrency and Computation: Practice and Experience*, vol. 15, no. 3-5, pp. 485–499, 2003.

[3] S. Khoshnood, M. Kusano, and C. Wang, "Concbugassist: Constraint solving for diagnosis and repair of concurrency bugs," in *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. ACM, 2015, pp. 165–176.

[4] M. Musuvathi and S. Qadeer, "Iterative context bounding for systematic testing of multithreaded programs," in *ACM Sigplan Notices*, vol. 42, no. 6. ACM, 2007, pp. 446–455.

[5] A. Bouajjani, M. Emmi, C. Enea, and J. Hamza, "Tractable refinement checking for concurrent objects," *Acm Sigplan Notices*, vol. 50, no. 1, pp. 651–662, 2015.

[6] A. Horn and D. Kroening, "Faster linearizability checking via p-compositionality," in *International Conference on Formal Techniques for Distributed Objects, Components, and Systems*. Springer, 2015, pp. 50–65.

[7] M. Pradel and T. R. Gross, "Fully automatic and precise detection of thread safety violations," *Acm Sigplan Notices*, vol. 47, no. 6, pp. 521–530, 2012.

[8] Z. Long and Y. Zhang, "Checking linearizability with fine-grained traces," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. ACM, 2016, pp. 1394–1400.

[9] S. Qadeer, A. Sezgin, and S. Tasiran, "Back and forth: Prophecy variables for static verification of concurrent programs," *Tech. Rep. MSR-TR-2009-142*, 2009.

[10] M. Vechev and E. Yahav, "Deriving linearizable fine-grained concurrent objects," *ACM SIGPLAN Notices*, vol. 43, no. 6, pp. 125–135, 2008.

[11] S. Burckhardt, C. Dern, M. Musuvathi, and R. Tan, "Line-up: a complete and automatic linearizability checker," in *ACM Sigplan Notices*, vol. 45, no. 6. ACM, 2010, pp. 330–340.

[12] J. M. Wing and C. Gong, "Testing and verifying concurrent objects," *Journal of Parallel and Distributed Computing*, vol. 17, no. 1-2, pp. 164–182, 1993.

[13] G. Lowe, "Testing for linearizability." PODC, 2015.

[14] M. Emmi, C. Enea, and J. Hamza, "Monitoring refinement via symbolic reasoning," in *ACM SIGPLAN Notices*, vol. 50, no. 6. ACM, 2015, pp. 260–269.

[15] J. E. Gottschlich, G. A. Pokam, C. L. Pereira, and Y. Wu, "Concurrent predicates: A debugging technique for every parallel programmer," in *Proceedings of the 22nd international conference on Parallel architectures and compilation techniques*. IEEE Press, 2013, pp. 331–340.

[16] C.-S. Park and K. Sen, "Concurrent breakpoints," in *ACM SIGPLAN Notices*, vol. 47, no. 8. ACM, 2012, pp. 331–332.

[17] S. Burckhardt, P. Kothari, M. Musuvathi, and S. Nagarakatte, "A randomized scheduler with probabilistic guarantees of finding bugs," in *ACM Sigplan Notices*, vol. 45, no. 3. ACM, 2010, pp. 167–178.

[18] M. Herlihy and N. Shavit, *The Art of Multiprocessor Programming, Revised Reprint*. Elsevier, 2012.

[19] A. Haas, T. Htter, C. M. Kirsch, M. Lippautz, M. Preishuber, and A. Sokolova, "Scal: A benchmarking suite for concurrent data structures," 2015.

[20] E. Ladan-Mozes and N. Shavit, *An Optimistic Approach to Lock-Free FIFO Queues*. Springer Berlin Heidelberg, 2004.

[21] S. Doherty, D. L. Detlefs, L. Groves, C. H. Flood, V. Luchangco, P. A. Martin, M. Moir, N. Shavit, and G. L. Steele Jr, "Dcas is not a silver bullet for nonblocking algorithm design," in *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*. ACM, 2004, pp. 216–224.

Predicate Interpretation Analysis Based on Soot

Chunrong Fang, Qingkai Shi, Yang Feng, Zicong Liu, Xiaofang Zhang, Baowen Xu*
State Key Laboratory for Novel Software Technology, Nanjing University
Nanjing 210093, China
*Email: bwxu@nju.edu.cn

***Abstract*—Symbolic execution maintains a path condition pc for every possible path of a program. It is challenging to construct a pc if some complex issues are involved in the path. A predicate interpretation *pi* is a subexpression of a pc and a pc of a path is a conjunction of all *pi*s in the path. Predicate interpretation has been widely used in theoretical analysis on domain testing and related fields. It recently emerges new impact on software testing by using partial path constraints to generate test data. In this paper, we propose an approach to produce *pi*s in a program. A tool for predicate interpretation analysis for Java programs is implemented based on the data-flow framework of Soot. Most of Java features can be handled in our tool. Moreover, a formal rule of predicate interpretation analysis is presented for more applications in the future. The experimental results show that our tool can produce *pi*s of a program effectively and efficiently.**

I. Introduction

Predicates play a key role in programs to fulfill functionalities with different execution paths. Predicate analysis on paths have shown increasing significance in software testing and debugging. Symbolic execution is a commonly used technique to construct path conditions (pc) as constraints, which hold the execution of a path in a program [7]. However, there are some challenges to make it practical in industry. One of the challenges is to effectively construct a pc of a path involving some complex issues in a program [2].

Predicate interpretation (pi) [10] is a subexpression of a pc. That is, a pi is a partial path constraint and a pc is a conjunction of all pis in the path. A pi is a predicate with only input variables. pis on a branch may be different because it may appear on different execution paths. pi has been widely used in theoretical analysis on domain testing and related fields. Recently, pi emerges new impact on software testing by using partial path constraints to generate test data, as pis are more practical than a pc involving some complex issues. This inspires us to build an effective tool for predicate interpretation analysis on modern programming languages, such as Java.

Data-flow analysis is an important method of program static analysis [6], which provides the theoretical basis of predicate interpretation analysis. Soot [8], a well-known framework for analyzing or optimizing Java byte-code, based on which many analysis tools have been implemented[1]. A key feature of the Soot framework is its excellent support for implementing intra-procedural data-flow analysis. In addition, Soot includes call-graph information and pointer analysis framework, which help deal with some complex issues in our approach.

[1]http://www.sable.mcgill.ca/soot/

In this paper, we propose an approach of predicate interpretation analysis based on data-flow framework of Soot. Our approach collects appropriate predicates respectively and replaces variables according to assignment statements along with the corresponding data flows. A novel feature of our approach is using loop induction to reduce the search space and improve the speed of analysis. A formal rule of predicate interpretation produced by the tool is presented for further uses. Additionally, a preliminary experiment was conducted. The experimental results show that our tool can produce pis of programs effectively and efficiently.

II. Approach

A. Data-flow Framework

In this section, we adopt a unified data-flow analysis framework based on lattice [6] to support PI analysis. The framework (D, V, $\wedge$, F) [1] is described as follows. (1) A direction of the data flow D, which is either FORWARDS or BACKWARDS. (2)A semi-lattice, including a domain of values V and a merge operator $\wedge$. A semi-lattice has a top element, denoted $\top$, such that for all $x \in V$, $\top \wedge x = x$. (3) A family F of transfer functions from V to V. A function in F, such as f_s, is a transfer function of the statement s.

We denote the data-flow values before and after each statement s by IN[s] and OUT[s], respectively. Given a data-flow graph and a backward data-flow problem, the unified algorithm [1] is shown in Figure 1.

```
initialize a value set: v;
IN [EXIT] = v;
for (each statement s other than EXIT) IN[s] = T;
while (changes to any IN occur)
      for (each statement s other than EXIT) {
            OUT[s] = ∧ m a successor of s IN[m] ;
            IN[s] = f_s(OUT[s]);
      }
```

Fig. 1. Algorithm of backward data-flow

Data-flow values of every point in the data-flow graph are initialized firstly. An iteration from the exit to the entry of the graph is followed by the initialization. Each time we pass by a statement s. The transfer function of the statement will be used. If two data-flow values meet at a point, they will be merged in accordance with the merge operation.

DOI reference number: 10.18293/SEKE2017-052

B. Predicate Interpretation

A predicate interpretation (pi) is a predicate that only contains parameters of a program [10]. If a predicate can be represented by $expression\Theta 0$, where Θ is a relational operator and $expression$ is a common arithmetic expression, the corresponding pi is $expression$ only with parameters $\Theta 0$.

PI analysis uses BACKWARDS data-flow technique. The value domain V is the set of predicates, and the merge operation $\wedge$ is union $\cup$. Some key points of data-flow framework for PI analysis are summarized in Table 1. It is a monotonic and distributive [6] data-flow framework which implies the solution of the problem is the maximum fixed point solution to the data-flow equations. We can apply these framework elements in Table 1 to the algorithm in Figure 1 to obtain the result of PI analysis.

TABLE I
PREDICATE INTERPRETATION ANALYSIS

Elements	PI Analysis
Domain	Set of predicates
Direction	Backwards
Transfer Function	(1)(2)(3)
Boundary	IN[EXIT]=Ø
Merge ($\wedge$)	$\cup$
Data-flow Equations	IN[s]= f(OUT[s]), OUT[s]= $\wedge_{p,succ(s)}$IN[p]
Initialize	IN[s]=Ø, for each s.

Transfer functions in Table 1 are important for PI analysis. We explain the transfer functions in detail as follows.

(1) If a statement s is a branch statement, the transfer function of s will be

$$f_s(x) = x \cup gen_s \tag{1}$$

where x is the pi set before the statement and gen_s is a set of predicates in the statement s.

(2) If a statement s is an assignment statement of variable v. Suppose the statement is $v = g(v, y_0, y_1, y_2, \cdots)$, where g is a function about variables v and $y_i, i = 0, 1, 2, \cdots$. The transfer function is

$$\forall pi(v) \in x,\ \ f_s(pi(v)) = pi(g(v, y_0, y_1, y_2, \cdots)) \tag{2}$$

which means variable v in every predicate in x, the pi set before statement s, will be substituted with $g(v, y_0, y_1, y_2, \cdots)$. In the equation, $pi(v)$ means a predicate containing the variable v.

(3) For others, transfer function is an identity function I about the domain value x, such that

$$I(x) = x \tag{3}$$

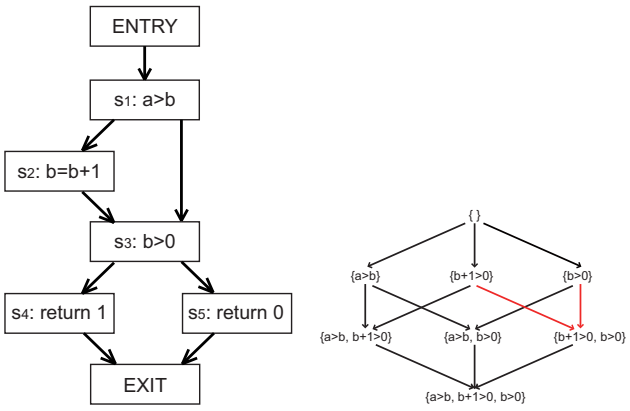
C. An Example

We introduce a simple example to explain our approach in detail. Figure 2(a) shows a control-flow graph.

If we define a partial order $\leq$ of a semi-lattice, such that

$$\forall x, y \in V,\ \ x \leq y\ iff.\ x \wedge y$$

the partial order $\leq$ of PI analysis semi-lattice will be $\supseteq$ because the merge operation is $\cup$. Following by the definition, we can draw the domain V as a lattice diagram in Figure 2(b).



(a) A control-flow graph (b) Lattice diagram

Fig. 2. An Example

The edges are all downward. From the diagram, we can get the result of merge operation easily.

Before the statement s_3: $b > 0$, OUT[s_5]=$\{\}$ due to the initial condition and using the identity transfer function. After s_3, IN[s_3]=f_{s_3}(OUT[s_3]) = $\{\} \cup \{b > 0\} = \{b > 0\}$. It uses the transfer function (1), and OUT[s_2]=OUT[s_1]=IN[s_3]=$\{b > 0\}$.

When analysis passes assignment statement s_2, using transfer function (2), we replace variable b with $b + 1$, such that IN[s_2] = $\{b+1 > 0\}$. IN[s_2] and IN[s_3] will meet at the exit of statement s_1, where we use the merge operation according to the red edges from Figure 2(b), such that, OUT[s_1]=IN[s_2] $\cup$ IN[s_3]= $\{b + 1 > 0, b > 0\}$.

Finally, we can obtain the resulting set of pis, which equals to IN[ENTRY]=$\{a > b, b + 1 > 0, b > 0\}$.

III. IMPLEMENTATION

The framework of our tool for PI analysis based on Soot is illustrated in Figure 3. It is made up of four main parts:(1) **Jimple Parser** transforms Java byte code to Jimple code, which is provided by Soot. (2) **Transformer** transforms the input Jimple code, with our two new transformers: `MethodsInliner` for inlining some user-defined methods. `StaticFieldInitializer` for initialize some static fields. (3) **Data-flow Analysis** is built on the data-flow framework of Soot. (4) **Filter** is used to filter some redundant expressions.

A. Transformer

Transformers accept Jimple codes of a program body as input, and then transform them by two new body transformers. The first one is `MethodsInliner`, which is used to inline methods. In default case, our tool will inline all user-defined methods and Java library methods will remain. The second one is `StaticFieldsInitializer`, which is used to initialize static fields of a class.

In most cases, `MethodsInliner` firstly inlines all user-defined methods, and obtain some classes whose static fields have been used in the program body.

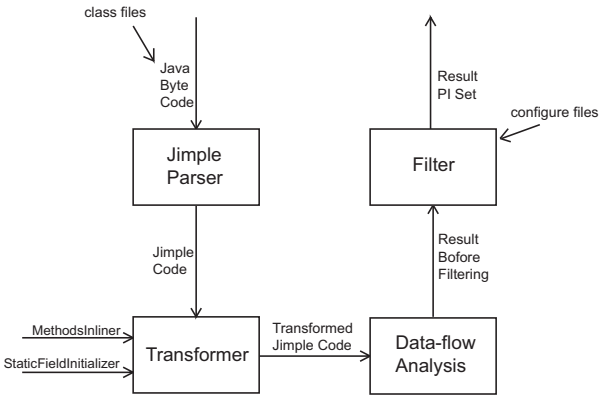


Fig. 3. Framework

Secondly, `StaticFieldsInitializer` adds methods `<clinit>()` of the classes, which are collected from the first step, to the head of program body. Lastly, `MethodsInliner` inlines those `<clinit>()` methods.

An example of a class called `Example` is shown in Figure 5. In the class, method `f` calls `g`, and `g` uses a static field of this class. With the last two transformations, the final predicate interpretation set is {`@parameter0 ≤ 10`}. Otherwise it will be {`@parameter0 ≤ <Example: int CONST>`}, which is not as elegant as the former.

```
int f(int){
   ......
   i0 := @parameter0: int;
   virtualinvoke
   r0.<Example: int g(int)>(i0);
   ......
}
```

(A) Before Transformation

```
int f(int){
   ......
   i0 := @parameter0: int;
   $i3 = <Example: int CONST>;
   if i0 <= $i3 goto label0;
      ......
   label0:
      ......
}
```

(B) After 1st MethodsInliner

```
int f(int){
   ......
   i0 := @parameter0: int;
   staticinvoke
   <Example: void <clinit>()>();
   $i3 = <Example: int CONST>;
   if i0 <= $i3 goto label0;
      ......
   label0:
      ......
}
```

(C) After StaticFiledsInitializer

```
int f(int){
   ......
   i0 := @parameter0: int;
   <Example: int CONST> = 10;
   $i3 = <Example: int CONST>;
   if i0 <= $i3 goto label0;
      ......
   label0:
      ......
}
```

(D) After 2nd MethodsInliner

Fig. 4. Transformation

Comparing similar primitive approach in Soot, `MethodsInliner` have two significant advantages, (1) `MethodsInliner` uses a recursion algorithm to inline all methods that need to be inlined in the program body. (2) `MethodsInliner` takes polymorphism into account. For example, there is an invoke statement like `child.<Super: void func()>()` where `child` is an instance of a class whose super class is `Super`. In this situation, we firstly get the type of `child`, and redirect the method to the child class.

B. Data Flow Analysis

Data flow analysis is the main part of our tool. We define `PIAnalysis` extending `BackwardFlowAnalysis` of Soot. The input of this part is a `TrapUnitGraph` of the transformed Jimple code which adds edges from every trapped unit to the trap's handler unit.

`PIAnalysis` overrides several key methods, four of which are: `flowThrough()` corresponding to Data-Flow Equation, `merge()` corresponding to Merge($\wedge$), `entryInitialFlow()` corresponding to Boundary, `newInitialFlow()` corresponding to Initialize in Table 1.

The most important method is `flowThrough()`, in which we handle fifteen kinds of statements of Jimple [9]. Most of them need not do anything, because they use the identity transfer function. However, three kinds of them should be dealt with by ourselves using the transfer function (1) and (2). The first is branch statements including `IfStmt` and `SwitchStmt`. The second includes `IdentityStmt` and `AssignStmt`. And another one is `InvokeStmt`. In addition, induction is used for loops to reduce search spaces and improve the speed of analysis.

1) Branch Statement: Every `IfStmt` consists of a predicate and a goto statement. The predicate will be added to the set of *pi*s according to the transfer function (1).

`SwitchStmts` have many branches. The tool creates a predicate for each value of switch statement. For instance, `switch(key){ case` v_1`: ⋯; case` v_2`: ⋯; ⋯}`, and {`key ==` v_1`, key ==` v_2`, ⋯`} will be added to the *pi* set.

2) Identity Statement and Assignment Statement: There is only one item in the two sides of the connector := in `IdentityStmt`. Given an identity statement which defines variable `i0` as the integer parameter, `i0 := @parameter0: int`, we use the transfer function (2) to replace an item with the other.

The most complex case is the `AssignStmt`. We must deal with many kinds of expressions on the two sides of =, such as `BinopExpr`, `ArrayRef`, `InvokeExpr`, etc. We must pay attention to two cases of them.

The first case is when the left side of = is an `ArrayRef`. In many earlier researches, arrays were considered as common objects whose fields were labeled with integer indexes. However, they ignored a big difference. Fields of a common object have fixed names, but if we consider an array as an object, the names of its fields will change at any time (e.g. modifying `array[i]` sometimes implies that `array[j]` is also changed if `i` equals to `j`). Figure 5(A) gives an example to explain how to deal with them. If we substitute `array[i]` with `6` according to the name, the condition expression will be `6 == 7`. However, if `i` equals to `j` during runtime, the condition expression will be `7 == 7`. Therefore, we keep the predicate instead of replacing the variable. For instance, `array[i]==7` will remain, and `i` can be considered as a reference variable whose value can be determined by some strategies (e.g. generating randomly, etc.) in further use.

The second case is when the right side of = is an `AnyNewExpr`. We keep these assignment statements to avoid confusion of different objects. Figure 5(B) presents an example. If we substitute `arr1` with `new int[2]`, we won't be able to distinguish `arr1` from `arr2`. In this example, the sequence {`arr1=new int[2], arr1[1]==7`} will remain.

```
func (int[] array, int i, int j){
    ......
    array[i] = 6;
    array[j] = 7;
    if (array[i] == 7) {
        ......
    }
    ......
}
```

(A) ArrayRef Example

```
func (){
    int[] arr1 = new int[2];
    int[] arr2 = new int[2];
    ......
    if (arr1[1] == 7) {
        ......
    }
    ......
}
```

(B) AnyNewExpr Example

Fig. 5. Assignment Statement

3) Invoke Statement: Most `InvokeStmts` have been inlined into the program body except Java library methods in default case. In most situations, inlining all methods is impossible and unnecessary, and some of the remaining methods may affect the values of local variables. These methods will be put into the result set with predicates, because in static analysis we can't judge whether they will change their arguments or callers. For example, a sequence {`aList.clear(), aList.isEmpty() == 0`} will be kept. With the invoke statement `clear()`, `aList.isEmpty() == 0` will be considered as an input-independent pi. On the other hand, if we can confirm a method is insignificant, such as `println(Object)`, we will put them in a configuration file so that the method can be filtered by textbfFilter.

4) Loops: Loop is an important issue which will lead to producing too many similar pis like $a > 0, a+1 > 0, a+2 > 0$ and so on, and even infinite data-flow analysis. To avoid these problems, we define some reference variables to help present loop variables contained in loop conditions.

A predicate of a simple loop like the one shown in Figure 6(A) can be simply written as $b + 4 * i < 10$ where i is a reference variable. If a loop variable is modified in more than one places in the loop body (e.g. Figure 6(B)), we will define several reference variables for it. Assume the loop in Figure 6(B) is executed i times and the first branch of the if statement is executed j times, then the other branch must be executed $i - j$ times. The predicate for the loop will be written as $a + 2 * j + 3 * (i - j) + 1 * i < 10$ where i and j are reference variables that can't be determined in static analysis. If we nest the simple loop in Figure 6(A) to the end of the loop in Figure 6(B), the pis of the nested loops will be $\{b+4*m*n < 10, a+2*j+3*(n-j)+1*n < 10\}$ where the simple loop executes m times, the outer loop executes n times, and the first branch of "if" executes j times.

In summary, induction for loop variables cannot only simplify the result set of pis, but also reflect the program structure more precisely.

```
while (b<10){
    b=b+4;
}
```

(A) A simple loop

```
while (a<10){
    a=a+1;
    if(...) a=a+2;
    else  a=a+3;
}
```

(B) A complex loop

Fig. 6. Loop

C. Filter

We obtain a set of pis after the data-flow analysis part. Some expressions in the pi set are redundant which should be removed. We implemented a filter to improve the results by filtering four kinds of expressions: (1) **Reduandante** pi**s** defined as Opposite predicates or same predicates. For example, $a > b$ and $b < a$ are the same predicate in semantics, while $a > b$ and $a \leq b$ are opposite, however, reflecting the same program structure. (2) **Input-independent** pi**s**, which are relations between constants, such as $2 == 3$ which must be false.(3) **Useless invoke statements**. For example, a method `set(Local)` may change a local variable, but a method `System.out.print()` is insignificant. The latter is "useless" and can be ignored. (4) **Useless assignment statements**. If a remaining assignment statement has nothing to do with pis in the result set, it can be deleted.

D. Output rule

It is important to make our tool work with different related tools, such as constraint solvers. In order to be convenient to transform results to other input format, we have established a grammar for the result expressions. Figure 7 shows the main part of the grammar where we define three result forms: (1) pi_stmt for pis. This is the main part of the final result set. (2) new_stmt for remaining assignment statements whose right side is `AnyNewExpr`. These expressions are used to distinguish different objects. It is explained in section 3.2.2. (3) $invoke\_stmt$ for remaining invoke statements which is described in section 3.2.3.

IV. Preliminary Experiment

We designed and conducted a preliminary experiment to verify our tool on two subject programs, `jtcas` and `ordset`, from SIR [3]. The common characteristics of the programs are a few of numerical or string inputs and simple data structure. It can help present the analysis results clearly.

The experiment was conducted in the default case which means Java library methods will not be inlined into the program body. A classic pi from the experiment is as follows

$$staticinvoke$$
$$<java.lang.Integer: \ int\ parseInt(java.lang.String)>$$
$$(@parameter0: \ java.lang.String[][11]) <= 0$$

where a Java library method, `parseInt()`, is kept. The first parameter of the analyzed program is a character string array.

```
stmt = pi_stmt | new_stmt | invoke_stmt

pi_stmt = expr rel_op expr;
rel_op = ">" | "<" | ">=" | "<=" | "==" | "!=";

new_stmt = local "=" any_new_expr;
any_new_expr = new_array_expr | new_multi_array_expr
                    | new_expr;

invoke_stmt = instance_invoke_expr | static_invoke_expr;
instance_invoke_expr = "instanceinvoke" immediate
                            ".<" method_signature ">(" expr_list ")";
static_invoke_expr = "staticinvoke"
                            "<" method_signature ">(" expr_list ")";

expr = immdiate |  arithmetic operation between expr

immediate = constant | local | param_ref | instance_field_ref
              | static_field_ref | array_ref | invoke_stmt;
param_ref = "@parameter" int_constant ":" type
instance_field_ref = immediate ".<" field_signature ">";
static_field_ref = "<" field_signature ">";
```

Fig. 7. Parts of the grammar for expressions

If the 11th element is parsed as an integer, the value will be greater than or equal to 0.

The `main` method in `jtcas` and a test driver method of `ordset` was analyzed. The `main` method in `jtcas` accepts a character string array as input. These command line arguments are transformed to integers and they are handled with complex logic to avoid traffic collision. The test driver method of `ordset` creates two ordered sets, and some operations, like union, adding elements, are completed on them.

The statistics results of the experiment are shown in Table 2. These pis have been divided into three types, each of which results in a markedly different effect on the domain boundary [10] in domain testing: Equalities($=$), Inequalities ($<$, $>$, $\leq$, $\geq$) and Non-equalities($\neq$).

TABLE II
EXPERIMENTAL RESULTS

statistic results	jtcas		ordset	
	#	percent	#	percent
Equalities	0	0%	7	2.8%
Inequalities	17	70.8%	189	75.0%
Non-equalities	7	29.2%	56	22.2%
Total	24	100%	252	100%
Redundant #	86	–	133	–
Preprocessing Time (ms)	330		610	
Analysis Time (ms)	60		430	

We can acquire some conclusions according to the statistic results as follows. (1) In-equalities ($<$, $>$, $\leq$, $\geq$) of a common program have a large proportion among all pis. The proportion is about 70%, even more. The final result pi set contains only a few equalities (=). (2) There are a large number of redundant pis before filtering. This result suggests that the filter part is necessary, and it can simplify the results greatly. (3) The time cost of PI analysis is collected to demonstrate the efficiency of our tool. It can produce all pis within several seconds for most common programs.

V. RELATED WORK AND DISCUSSION

(pi) can be considered as a simplified version of path condition. pi focuses on the local structure information instead of the global. The compromise makes it more practical, especially in domain testing [10]. Jeng [4] integrated domain testing and data-flow testing which applied the pi to def-use chains in data-flow testing. Jeng and Weyuker [5] proposed a simplified domain testing strategy which was not limited to linear pis any longer. In addition, Zhao [11] presented a method to generate domain boundary for character strings.

PI analysis can also be done by forward data-flow analysis called symbolic analysis [1] which is regularly used in traditional optimizing compilers. The main idea of symbolic analysis is using parameters as reference variables to present local variables in the program body. However, if it is used in our approach, many resources will be wasted to analyze unconcerned variables, which may not be used in predicates later. Thus, our approach adopts backward data-flow analysis to get predicates firstly, and then analyze the related variables without any waste.

VI. CONCLUSION

In this paper, we introduce an approach for predicate interpretation analysis based on data-flow analysis. Besides, we implement a predicate interpretation analysis tool, which incorporates loop induction to reduce the search space and improve the speed of analysis. Further we conducted an experiment and the results show that our tool can produce pis of programs effectively and efficiently.

ACKNOWLEDGEMENT

This work was partially supported by the National Basic Research Program of China (973 Program 2014CB340702), the National Natural Science Foundation of China (No. 61373013, 61170067), HPI and HPI Research School, Program B for Outstanding PhD Candidate of Nanjing University.

REFERENCES

[1] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman. *Compilers–Principles, Techniques, and Tools*. Addison-Wesley Publishing Company, 2006.

[2] C. Cadar, P. Godefroid, and etc. Symbolic execution for software testing in practice–preliminary assessment. In *ICSE'11*, pages 1066–1071, 2011.

[3] G. R. Group et al. Software-artifact infrastructure repository (sir), 2009.

[4] B. Jeng. Toward an integration of data flow and domain testing. *Journal of Systems and Software*, 45(1):19–30, 1999.

[5] B. Jeng and E. J. Weyuker. A simplified domain-testing strategy. *ACM Transactions on Software Engineering and Methodology*, 3(3):254–270, 1994.

[6] G. Kildall. A unified approach to global program optimization. In *PLDI'73*, pages 194–206, 1973.

[7] J. C. King. Symbolic execution and program testing. *Communications of the ACM*, 19(7):385–394, 1976.

[8] R. Vallee-Rai, P. Co, E. Gagnon, and etc. Soot–a java bytecode optimization framework. In *CASCON'99*, pages 125–135, 1999.

[9] R. Vallee-Rai and L. J. Hendren. Jimple: Simplifying java bytecode for analyses and transformations. Technical report, 1998.

[10] L. J. White and E. I. Cohen. A domain strategy for computer program testing. *IEEE Transactions on Software Engineering*, 6(3):247–257, 1980.

[11] R. Zhao, M. R. Lyu, and Y. Min. Domain testing based on character string predicate. In *ATS'03*, pages 96–101, 2003.

Generating Software Agents for Data Mining: An Example for the Health Data Area

Reinier Morejón[1], Marx Viana[1], Carlos Lucena[1]
[1]Laboratory of Software Engineering (LES) - Pontifical Catholic University - PUC-Rio
Rio de Janeiro, RJ - Brazil
{rnovales, mleles, lucena}@inf.puc-rio.br

Abstract— Data mining has been a hot topic that attracts both database and machine learning researchers. Due to the constant growth of data volume there is an increasing need to obtain knowledge from these large data sets that are very difficult to handle and process with traditional methods. Software agents can play a significant role performing data mining processes in ways that are more efficient. For instance, they can work to perform selection, extraction, preprocessing and integration of data as well as parallel, distributed, or multisource mining. This paper proposes a framework based on multi-agent systems to apply data mining techniques to health data sets. For a first usage scenario, we use a data set for hypothyroidism and run two mining processes in parallel.

Keywords. Multi-Agent Systems, Data Mining, Machine Learning

I. INTRODUCTION

Since its origins, computing science has always represented an accelerated path of constant evolution and transformation, at the same time implying a similar progression of information generation. Not only is this evidenced by the spectacular growth of traditional Internet usage [21] and the equally outstanding rise of the Internet of Things [26] [9], but also by the large volume of data generated by the healthcare industry [25].

Health care services are an expanding source of vast quantities of information about patients' health. Beyond the traditional approaches of clinical practice, databases in today's modern health institutions automatically collect "structured data relating to all aspects of care" [17] such as "diagnosis, medication, test results and radiological imaging data" [17]. These phenomena have a fundamental underlying element: the huge volume of data in constant expansion - data that can vary in types and sources [17], and need to be mined to produce significant knowledge. A task like this is very difficult to perform in traditional ways. Here is where Data Mining plays its role.

There are sources such as: (i) legacy systems like devices attached to, or inside, the human body to monitor and maintain human health and wellness; (ii) disease management and fitness tracking [26], and (iii) huge data sets of medical records for research studies [12]. These sources represent an astonishing volume of data available. Obtaining meaningful knowledge is very important in this area. As Durairaj said in [10], the application of data mining algorithms in the healthcare industry plays "a significant role in prediction and diagnosis of diseases."

Although there are already some solutions using data mining to deal with specific health data - for example, diabetes [12] or heart disease [24], it is unusual to see architectural models designed for knowledge discovery from medical data records that have been applied to more than one disease. Adjustable solutions can adapt from one disease to another simply by setting new instances. We can cite hypothyroidism [15] and its complications [7], which brings an extensive possibility of research. Therefore, in this research we will be mining a hypothyroidism data set.

In this context, we present the framework Java Agent Framework for Health Data Mining (JAF4HDM). This framework aims at providing support to build and operate different agents to run classification-mining methods. In order to take advantage of software agents to better implement data mining techniques over health data systems, such framework extends the JADE framework [2][3], which already provides support to building autonomous and pro-active agents. By using these new features, it is possible to build agents for classification-mining methods to: (i) work on one data set, (ii) perform the training process, and (iii) predict new unlabeled instances by using the rules obtained as a result of the training process.

This paper is organized as follows: Section II presents a background about the relationship between Data Mining, Machine Learning and Multi-Agent Systems. Section III has a brief background of WEKA. Section IV presents the JADE framework. Section V discusses some related work. Section VI presents JAF4HDM. Section VII describes a case study by showing how agents help to classify hypothyroidism data. Section VIII presents our conclusion and future work.

II. THE RELATIONSHIP BETWEEN DATA MINING, MACHINE LEARNING AND MULTI-AGENT SYSTEMS

From the onset five important trends have been present [29] in computing: ubiquity, interconnection, intelligence, delegation and human-orientation. These trends have grown at a steady pace, which means more processing power; distributed and big scale architectures; more automation, and constant evolution for new solutions in the computing field. Likewise, the volume of information generated by computing systems has grown consistently and overwhelmingly.

The volume of data in the world and in our lives seems to increase steadily—and there is no end in sight [28]. Nowadays, information technologies (ITs) overwhelm us with data as ITs systems record every action we take when we use

DOI reference number: 10.18293/SEKE2017-086

them. This happens not only with personal actions but also with all kinds of decisions and operations that take place in the world of commerce and industry [28]. In addition, the huge volume of data that health care transactions generate are too complex and voluminous to be processed and analyzed by traditional methods [18].

Throughout recent decades, the magnitude of data volume and ubiquity, for example, has resulted in two of the most prominent, dynamic, and exciting research areas in information sciences: (i) autonomous agents and multi-agent systems (MASs), and (ii) data mining and knowledge discovery in databases (KDD) [8].

A software agent is a computer system situated in some environment, capable of autonomous action in order to meet its design objectives [29]. In other words, an agent can figure out by itself what needs to be done without the need to receive an explicit order of what to do at any given moment.

A multi-agent system is one that consists of a number of agents, which interact with one another, typically by exchanging messages through some computer network infrastructure. To interact successfully, these agents must be able to cooperate, coordinate, and negotiate between themselves [29].

On the other hand, we have data mining techniques that have different objectives from those of multi-agent systems. As in [28], data mining is the process of discovering patterns in data. The process must be automatic or (more usually) semiautomatic. The patterns discovered must be meaningful in that they lead to some advantage—usually economic.

Experience shows that in many data mining applications, the explicit knowledge structures acquired and the structural descriptions are as important as—and often much more important than—the ability to perform well on new examples [28]. Then, if MAS and data mining have evolved with well-defined, yet distinct, aims and objectives [8], how are they related?

Typical problems in agents that could find satisfactory solutions in data mining include multi-agent learning, adaptation, evolution, and behavior analysis. For instance, knowledge extracted through data mining could provide more stable, predictable, and controllable models for dispatching and planning, or it can assist in the self-organization and evolution of multi-agent systems in acceptable directions. On the other hand, software agents can support and enhance the knowledge discovery process in many ways. For example, agents can contribute to data selection, extraction, preprocessing, and integration and they are an excellent choice for parallel, distributed, or multisource mining [8].

III. WEKA

Waikato Environment for Knowledge Analysis (WEKA) is a collection of machine learning algorithms for data mining tasks. WEKA provides several implementations of learning algorithms to apply to data sets. Furthermore, it includes a diversity of tools for transforming data sets; it also allows preprocessing a data set, feeding it into a learning scheme, and analyzing the resulting classifier and its performance—all without writing any program code at all [11].

WEKA is easy to extend thanks to simple API and plug-in mechanisms and facilities that automate the integration of new learning algorithms by means of its graphical user interface [16].

The workbench includes methods for the main data mining problems: regression, classification, clustering, association rule mining, and attribute selection. In addition, it provides many data visualization facilities and data preprocessing tools. All algorithms take their input in the form of a single relational table obtained from a file or generated by a database query [11].

There are three ways of using WEKA: (i) to apply a learning method to a data set and analyze its output to learn more about the data; (ii) to use learned models to generate predictions on new instances (See Section VII), and (iii) to apply several different learners and compare their performance in order to choose one for prediction [11].

For a Java environment, it is possible to solve a learning problem without writing any machine learning code, just by accessing the available WEKA algorithms [11].

IV. JADE

The Java Agent Development Framework (JADE) is a software framework to support the development of agent applications in compliance with the FIPA 2000 specifications for interoperable intelligent multi-agent systems [6]. Fig. 1 shows a standard model of an agent platform, according to FIPA.

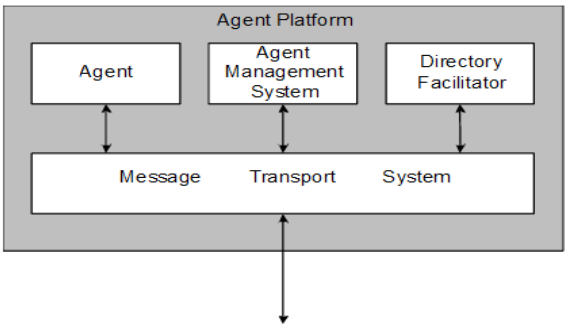


Figure 1. Reference architecture of a FIPA Agent Platform [3].

According to Fig. 1, three elements are essential in a FIPA compliant agent platform. First, the Agent Management System (AMS)—only one per platform—which is the agent that exerts supervisory control over access to, and use of, the Agent Platform. Each agent must register with an AMS in order to get a valid agent identifier (AID) [3]. Second, the Directory Facilitator (DF), the agent that provides the default yellow page service in the platform. Finally, the Message Transport System, (or Agent Communication Channel (ACC)), the software component controlling all the exchange of messages within the platform, including messages to/from remote platforms.

There are two different points of view for describing the JADE system: first, JADE is a runtime system for FIPA

compliant Multi-Agent Systems, supporting application agents every time they need to exploit some FIPA covered feature such as the life-cycle management of the agent. Second, JADE is also a Java framework for developing FIPA-compliant agent applications, making FIPA standard assets available to the programmer through object-oriented abstractions [6].

JADE includes both the libraries (i.e. the Java classes) required for developing application agents, and the runtime environment that provides the basic services. This environment must be active on the device before the agent's execution moment. Each instance of the JADE run-time is called container. All containers taken as a whole form the platform, which provides a homogeneous layer that hides the complexity and the diversity of the underlying tiers (hardware, operating system, type of network, JVM) from agents and application developers [5].

V. Related Work

Some approaches proposed [12, 8, 14] perform the data mining process by using Multi-Agent Systems. For instance, in [14], the authors proposed a MAS Technology for Distributed Data Mining (DDM) and Distributed Classification (DC). They addressed as a key problem the fact that data sources are distributed, heterogeneous and, as a rule, of large scale [14]. In addition, the authors addressed as an issue that the Distributed Data Mining MAS design technology presumes collaborative activities of agent-mediated distributed users [14]. Therefore, they proposed an architecture of DDM and DC MASs and a technology for their design. They further proposed a number of well-developed protocols supporting agent-mediated design of applied DDM and DC MASs [14].

Gao, Denzinger and James [12] proposed a cooperative multi-agent data mining model to apply to medical data on diabetes. They developed CoLe, a model for cooperative agents for mining knowledge from heterogeneous data [12]. They designed that model having in mind the use of various agents to perform a data mining process over a completely heterogeneous data set by taking advantage of the cooperation between these agents to generate more significant and more useful structures of knowledge that no individual mining agent can produce alone [12]. This model works by making iterations of the mining process, and for each iteration, the discovered knowledge can be useful for the next one. Based on the proposed model, the authors implemented, and tested, a multi-agent system for mining diabetes related data, by having two agents running mining algorithms and another agent with methods to produce hybrid rules. As a result of the experiments that were carried out, the implemented system outperformed the individual mining algorithms, with better and high-quality rules. Another discovery was the confirmation of a close relation between diabetes and hypertension. An improvement to this work could be the creation of a generic framework according to the CoLe model. The authors implemented a solution to diabetes-related heterogeneous data sets, making it clear in [13] that the CoLe work is specific to Early Diabetes Detection.

The authors in [8] presented the synergies between Data Mining and Software Agents, creating the concept of Agent Mining. They generated a Disciplinary Framework - "a high-level research map of agent mining as a disciplinary area" [8]. They also gave an overview of what agent mining means: "a young scientific field that advocates and studies the integration of agent technology and data mining" [8]. In addition, they performed a deep analysis of agent technology and data mining, speaking about the advantages and flaws of each and about the unquestionable benefits of them working together. In short, the authors show that the potential of "agent mining for the mutual enhancement of both fields and for the creation of super-intelligent system" [8] is promising and optimistic.

Agent Academy (AA) [23] is an open-source framework and an integrated development environment to create Software Agents and Multi-Agent Systems. The authors developed by using JADE and WEKA APIs. The framework provides embedded intelligence to agents by inserting decision models generated by a previous data mining process over a background data-specific application. The authors applied AA only to a supply chain management industrial scenario. There is no evidence about the applicability of AA to a health-oriented scenario.

VI. JAF4HDM: A Software Framework for Health Data Minig

This section describes the main elements required to understand the framework proposed in this paper. In addition, we will provide an overview of the JAF4HDM framework and discuss the different components, including the kernel (frozen-spots) and flexible points (hot-spots) [20].

A. The Framework

The JAF4HDM framework is implemented using software agents, as illustrated in Fig. 2. The JAF4HDM extends JADE [6], a FIPA compliant framework to implement Multi-Agent Systems developed in Java. In addition, it makes use of WEKA API algorithms to process health data. The framework comprises three main functions: (i) a workover of different data sources to preprocess the data and make it ready for training, (ii) a training process, by executing a desired mining method for the previously prepared data, and (iii) the execution of the prediction task, performed by agents using the rules generated by the training process over new unlabeled instances.

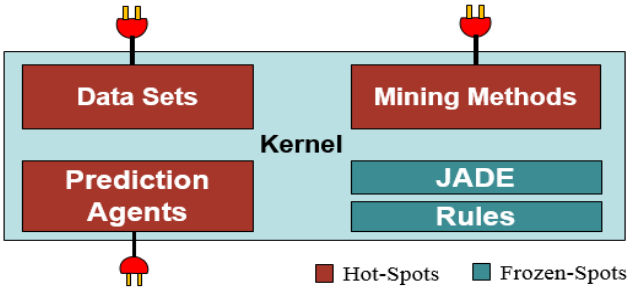


Figure 2. JAF4HDM framework architecture.

B. *The JAF4HDM Architecture*

The JAF4HDM framework supports the creation of a health data mining process that shows how to classify a given data set, such as hypothyroidism, which we show in Section VII.

We based the JAF4HDM mining process on the model proposed in [12] nuanced with some conceptual elements proposed by [30], with the difference that we ran several mining methods on a single homogeneous data set by making predictions based on each method-generated rules instead of having an agent combining all the generated rules in one single set of rules. In addition, our idea is to apply our solution to several illnesses instead of just one.

Fig. 3 shows that the process begins with one agent that prepares a single, homogeneous data set for training. Then, several agents will train different mining methods over that data set, obtaining a trained model with generated rules for each method. Finally, by using these rules other agents will predict the classification of each instance in a provided unlabeled data set.

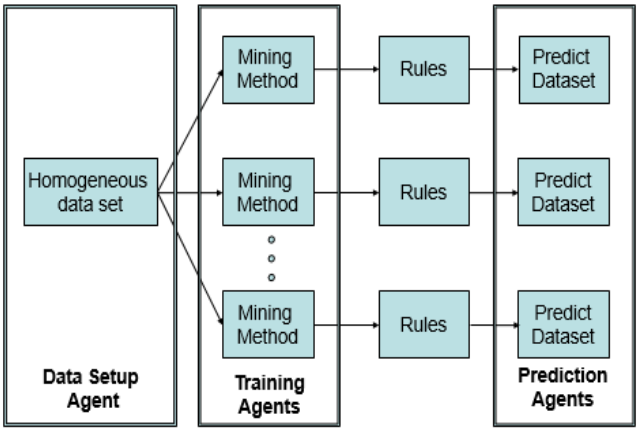


Figure 3. JAF4HDM mining process.

There are three kinds of agents (See Fig. 3) as part of JAF4HDM, described in Table 1:

TABLE I. AGENTS DESCRIPTION

Name	Description
DataSetupAgent	Prepares the data that has to be processed for training by selecting the source of the data, extracting the data and getting it ready for the training process.
TrainingAgent	Over the previous prepared data, and with a selected classifier algorithm, this agent will train and save a model, so that it can be used in future predictions.
PredictionAgent	This agent will take an unlabeled data set and will classify it with an existing trained model.

C. *Hot-Spots and Frozen-Spots*

As previously mentioned, JAF4HDM is an extension of the JADE framework, and therefore, they share the same core and hot-spots. The process used for the communication between software agents, and the identifiers of agents are examples of hot-spots that JAF4HDM inherited from JADE. In addition, JAF4HDM defines other specific hot-spots, which are:

Data Sets (*DataSetupAgent* Class): There are several ways to interact with a data set, from the format of the source (data base, csv file, arff file, etc) to the way that data is prepared (divided by cross validation split, or by percentage split). An agent will perform this task to make it more dynamic.

Mining Methods (*TrainingAgent* Class): There are many methods to carry out data mining - all different in some ways. Each instance must use different algorithms to teach their models. *TrainingAgents* will train and generate models by performing mining methods. In this first stage of the work, we use WEKA API to perform the algorithms; in the next stages, JAF4HDM will be able to use other APIs.

Prediction Agents (*PredictionAgent* Class): They must produce knowledge, but the ways to achieve this can differ depending on the selected methods. Consequently, these agents can conduct the new knowledge generation process in different ways.

The frozen-spots in this context (also named Kernel) are composed of: (i) the JADE framework; (ii) classes that work with the mining methods, and (iii) a learned set of rules saved in models to make subsequent predictions. Fig. 4 shows the class diagram of JAF4HDM, and its hot- and frozen-spots.

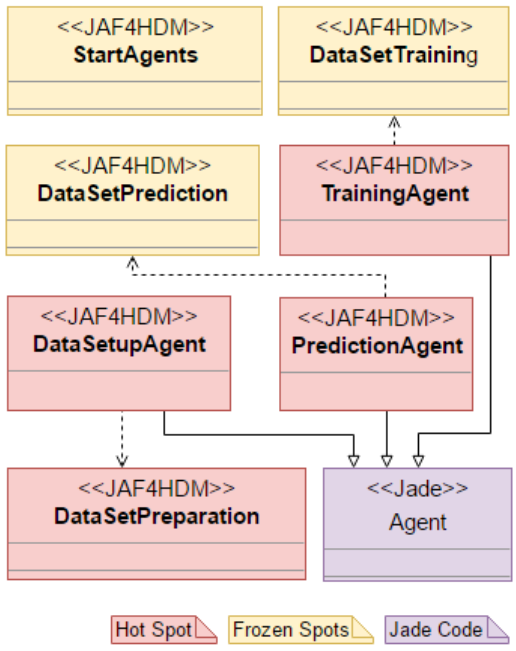


Figure 4. JAF4HDM Class Diagram.

To implement and deploy one application, based on the JAF4HDM framework, we need to implement at least one instance of each defined agent in order to successfully execute

the mining process in the usage scenarios. To build an application based on JAF4HDM, there must be at least one *DataSetupAgent*, one *TrainingAgent* and one *PredictionAgent* for an effective execution of the mining process.

VII. USAGE SCENARIO: MINING ON HYPOTHYROIDISM DATA

A framework is classified according to its extensibility; it can be used as a white box or a black box [20]. In white box frameworks, an instantiation is only possible through the creation of new classes. A black box framework allows a developer to produce new instances using configuration scripts. This type of framework automatically creates the classes and source code for the chosen configuration, which facilitates the instantiation process [20]. For now, in its first version, JAF4HDM is only a white box framework.

To test the framework applicability, it is necessary to implement one instance, which in this case is a specific scenario to predict unlabeled instances of hypothyroidism related data.

To clarify it, the new system first will train two models, using PART and Decision Table classification algorithms (available in WEKA API) by mining a data set of Thyroid disease records. The Garavan Institute and J. Ross Quinlan, New South Wales Institute, Sydney, Australia supplied this data set [27]. Then, with the trained model, we will predict the classification of each instance in another data set in the same format as the first one, but unlabeled.

Next, an explanation about how the framework in the current usage scenario was instantiated. The steps are as follows:

A. Put both *arff* file data sets (labeled to train and unlabeled to predict) in the root folder of the project. As a good practice, both the labeled and unlabeled filenames are almost the same, only differentiated by the string "-test" added at the end of the unlabeled data set filename.

B. The current version is for running one or more mining methods over a single data set, Therefore, to put in context the application with the right data set, two attributes must be set up manually with default values. These are: (i) *dataset_name* in *DataSetupAgent* Class with the name of the labeled data set, and (ii) *datasettest_name* in *PredictionAgent* Class with the name of the unlabeled data set.

C. We built the current JAF4HDM instance to process the data set by using two classifier algorithms PART and Decision Table from WEKA. Therefore, it is mandatory for each algorithm to extend the *TrainingAgent* Class and the *PredictionAgent* Class.

D. For each extended *TrainingAgent*, it is required to set the variables *modeltotrain* and *receiver*, which means, respectively, the algorithm to use and the name of the extended prediction agent that will use its results as entry.

E. For each extended *PredictionAgent*, it is required to set the variables *predictmodel* and *datasettestname*, which means, respectively, the same algorithm used by the extended *TrainingAgent* (whose results served as entry) and the name of the unlabeled data set to be predicted.

F. The agents must be defined in *StartAgents* Class before execution.

During and after the execution process, the system generates useful information such as the obtained rules after each training agent execution. Table 2 shows a fragment of the rules generated by the training agent that runs the PART algorithm over a hypothyroidism related data set. The system also generates a file containing the model obtained after the training process and a file containing a new data set already labeled as a result of the prediction process.

TABLE II. SAMPLE OF RULES GENERATED BY THE TRAINING AGENT THAT RUNS THE PART ALGORITHM

1	(TSH ≤ 6) → negative
2	((FTI ≤ 64) ∧ TSH_measured ∧ T4U_ measured ∧ ¬thyroid_surgery ∧ (T3 ≤ 2.3)) → primary_hypothyroid
3	(¬on_thyroxine ∧ TSH_measured ∧ ¬thyroid_surgery ∧ (TT4 ≤ 150) ∧ (TT4 > 48) ∧ TT4_measured ∧ (T3 ≤ 2.3))→ compensated_hypothyroid
4	on_thyroxine → negative
5	¬TSH_measured → negative
6	(thyroid_surgery ∧ (TT4 > 69)) → negative

As a sample of what a rule means in the data set context, we have in Table 2, line 1, the first generated rule, "(TSH ⩽ 6) → negative." For one instance that means that if the TSH (Thyroid Stimulating Hormone) measurement is less than or equal to 6, then the instance is classified as negative for hypothyroidism. The second line shows that if one instance has an FTI (Free Thyroxine Index) measurement less than or equal to 64 and both TSH and T4U (it is a test to measure Thyroxine) have been measured, and there has not been a thyroid surgery, and the T3 (Triiodothyronine) measurement is less than or equal to 2.3, then the instance is positive for primary hypothyroid. The same applies to the next rules. We use these learned rules after training to predict or classify new instances in real time. Table 3 shows the result of a prediction task for one hypothyroid-related unlabeled data set.

TABLE III. RESULTS AFTER PREDICTING UNLABELED DATA SET INSTANCES

Category	**Amount**
Total number of instances:	19
Instances classified as 'negative'	13
Instances classified as 'compensated hypothyroid'	4
Instances classified as 'primary hypothyroid'	2
Instances classified as 'secondary hypothyroid'	0

Table III shows how a prediction agent classifies each element of the unlabeled data set. For example, we demonstrate the prediction according to the use of one already trained model (based on PART algorithm) over one unlabeled data set containing 19 instances of hypothyroid-related data.

Out of 19 instances, the prediction classified 13 as negative, 4 as compensated hypothyroid, 2 as primary hypothyroid and none as secondary hypothyroid.

VIII. Conclusion and Future Work

This paper proposes the JAF4HDM framework for Health Data Mining, with the objective of providing support to build and operate different agents to run classification-mining methods to: (i) work on one data set, (ii) conduct the training process, and (iii) predict new unlabeled instances by using the rules obtained as a result of the training process. We can verify the applicability of the JAF4HDM framework by using the scenario presented in Section VII, where instantiated agents are responsible for running a mining process over a hypothyroidism data set to predict new instances later.

For future work our aim is to build features where: (i) the framework must be able to work with heterogeneous data sets using different methods on different types of data. It is necessary to conduct studies about heterogeneous data sets; (ii) the *DataSetupAgent* (See Table I) must be able to manage dynamically the extraction and preparation of data from its source in order to deliver to each *TrainingAgent* the adequate data set to perform the training process and, thus, avoiding to run on the whole data set directly, and (iii) The prediction process must have one agent that can receive the learned models from each training agent, find the way to mix the rules of these models, evaluate them and put the better ones into a final model. This agent will obtain meaningful information from models and will provide a feedback to trainers to improve their own work. In addition, it will help the *DataSetupAgent* to form the slices of data to process in the next iteration.

References

[1] BAOYUN, Wang. Review on internet of things [J]. Journal of electronic measurement and instrument, 2009, vol. 12, p. 1-7.

[2] BELLIFEMINE, Fabio, et al. Jade administrator's guide. TILab (February 2006), 2003.

[3] BELLIFEMINE, Fabio, et al. Jade programmer's guide. Jade version, 2002, vol. 3.

[4] BELLIFEMINE, Fabio Luigi; CAIRE, Giovanni; GREENWOOD, Dominic. Developing multi-agent systems with JADE. John Wiley & Sons, 2007.

[5] BELLIFEMINE, Fabio, et al. JADE: A software framework for developing multi-agent applications. Lessons learned. Information and Software Technology, 2008, vol. 50, no 1, p. 10-21..

[6] BELLIFEMINE, Fabio; POGGI, Agostino; RIMASSA, Giovanni. JADE: a FIPA2000 compliant agent development environment. Proceedings of the fifth international conference on Autonomous agents. ACM, 2001. p. 216-217.

[7] BERBER, Eren. Complications Of Hypothyroidism. EndocrineWeb, 2016. Web. 20 Feb. 2017.

[8] CAO, Longbing; GORODETSKY, Vladimir; MITKAS, Pericles A. Agent mining: The synergy of agents and data mining. IEEE Intelligent Systems, 2009, vol. 24, no 3.

[9] COLUMBUS, Louis. Roundup Of Internet of Things Forecasts And Market Estimates, 2015. Forbes, December, 2015, vol. 27.

[10] DURAIRAJ, M.; RANJANI, V. Data mining applications in healthcare sector: a study. Int. J. Sci. Technol. Res. IJSTR, 2013, vol. 2, no 10.

[11] FRANK Eibe , et al. The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques," Morgan Kaufmann, Fourth Edition, 2016.

[12] GAO, Jie; DENZINGER, Jörg; JAMES, Robert C. A cooperative multi-agent data mining model and its application to medical data on diabetes. International Workshop on Autonomous Intelligent Systems: Agents and Data Mining. Springer Berlin Heidelberg, 2005. p. 93-107.

[13] GAO, Jie; DENZINGER, Jörg; JAMES, Robert C. CoLe: A cooperative data mining approach and its application to early diabetes detection. Data Mining, Fifth IEEE International Conference on. IEEE, 2005. p. 4 pp.

[14] GORODETSKY, Vladimir; KARSAEYV, Oleg; SAMOILOV, Vladimir. Multi-agent technology for distributed data mining and classification. Intelligent Agent Technology, 2003. IAT 2003. IEEE/WIC International Conference on. IEEE, 2003. p. 438-441.

[15] HALL, Reginald; SCANLON, Maurice F. Hypothyroidism: Clinical features and complications. Clinics in Endocrinology and metabolism, 1979, vol. 8, no 1, p. 29-38.

[16] HALL, Mark, et al. The WEKA data mining software: an update. ACM SIGKDD explorations newsletter, 2009, vol. 11, no 1, p. 10-18.

[17] JENSEN, Peter B.; JENSEN, Lars J.; BRUNAK, Søren. Mining electronic health records: towards better research applications and clinical care. Nature Reviews Genetics, 2012, vol. 13, no 6, p. 395-405.

[18] KOH, Hian Chye, et al. Data mining applications in healthcare. Journal of healthcare information management, 2011, vol. 19, no 2, p. 65.

[19] KUMAR, Sujeet; KUMAR, Utkarsh. Java agent development framework. International Journal Research, 2014, vol. 1, no 9, p. 1022-1025.

[20] MARKIEWICZ, Marcus Eduardo; DE LUCENA, Carlos JP. Object oriented framework development. Crossroads, 2001, vol. 7, no 4, p. 3-9.

[21] MINIWATTS Marketing Group. Internet World Stats: Usage and population statistics. Retrieved from Internet World Stats: http://www. internetworldstats.com/stats.htm, 2016.

[22] MITCHELL, Tom M. Machine learning and data mining. Communications of the ACM, 1999, vol. 42, no 11, p. 30-36..

[23] MITKAS, Pericles A., et al. A framework for constructing multi-agent applications and training intelligent agents. International Workshop on Agent-Oriented Software Engineering. Springer Berlin Heidelberg, 2003. p. 96-109.

[24] PALANIAPPAN, Sellappan; AWANG, Rafiah. Intelligent heart disease prediction system using data mining techniques. Computer Systems and Applications, 2008. AICCSA 2008. IEEE/ACS International Conference on. IEEE, 2008. p. 108-115..

[25] RAGHUPATHI, Wullianallur. Data mining in healthcare. Healthcare Informatics: Improving Efficiency through Technology, Analytics, and Management, 2016, p. 353-372.

[26] SZEWCZYK, Paweł. Impact of the Internet of Things on the economy and society. Zeszyty Naukowe. Organizacja i Zarządzanie/Politechnika Śląska, 2016, no 93, p. 461-470.

[27] UCI KDD archive. Irvine, University of California, Department of Information and Computer Science, http://kdd.ics.uci.edu. Last access January 2017.

[28] WITTEN, Ian H., et al. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2016.

[29] WOOLDRIDGE, Michael. An introduction to multiagent systems. John Wiley & Sons, 2009.

[30] BRIOT, Jean-Pierre; DE NASCIMENTO, Nathalia Moraes; DE LUCENA, Carlos José Pereira. A multi-agent architecture for quantified fruits: Design and experience. En 28th International Conference on Software Engineering & Knowledge Engineering (SEKE'2016). SEKE/Knowledge Systems Institute, PA, USA, 2016. p. 369-374.

Distributed API Protocol Mining

Deng Chen[1, a], Yanduo Zhang[1, b], Wei Wei[1, c], Rongcun Wang[2, d], Xiaolin Li[1, e], Shixun Wang[3, f], Rubing Huang[4, g]

[1] Hubei Provincial Key Laboratory of Intelligent Robot, Wuhan Institute of Technology, Wuhan, P.R. China

[2] School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, P.R. China

[3] School of Computer and Information Engineering, Henan Normal University, Xinxiang, P.R. China

[4] School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang, P.R. China

[a] dchen@wit.edu.cn

[b] zhangyanduo@hotmail.com

[c] weiwei@huawei-elec.com

[d] rcwang@hust.edu.cn

[e] 932233986@qq.com

[f] wsxun@hust.edu.cn

[g] rbhuang@mail.ujs.edu.cn

***Abstract*—Dynamic Protocol Mining (DPM) techniques are a promising approach to infer useful API protocols automatically. However, their results are biased to input test cases and the instrumentation overhead discounts their usability in industrial practice. In this paper, we propose a distributed dynamic protocol mining framework NSpecMiner. Our framework is based on a client-server architecture, where the client tracer gathers Program Execution Traces (PETs) and sends them to the server for mining. Mined protocols are saved on the server to provide various kinds of remote services, such as API protocol retrieval and program verification, etc. Compared with local miners, NSpecMiner has many advantages: 1) A large number of diverse PETs are likely to be collected from multiple clients, which is essential for mining accurate and complete API protocols. 2) Instrumentation overhead can be balanced among multiple clients. 3) Via integrating the client tracer into widely used software, we can mine API protocols transparently and automatically without any human effort. To evaluate our technique, we performed a comparison test with a local miner ISpecMiner and NSpecMiner. Preliminary results show that our approach is effective to mine useful API protocols as local miners. While our method is able to gather PETs concurrently from multiple clients and other merits of the distributed technology will further benefit DPM significantly.**

Keywords—program temporal specification; program dynamic analysis; distributed specification mining; API protocol mining

I. INTRODUCTION

API protocols specify temporal constraints regarding the order of calls of API methods. For example, calling peek() on java.util.Stack without a preceding push() gives an EmptyStackException, and calling next() on java.util.Iterator without checking whether there is a next element with hasNext() can result in a NoSuchElementException. API clients that violate such protocols do not obtain the desired behaviors and may even crash the program [1].

API protocols are beneficial for many tasks of software development, such as program documentation, understanding, testing, verification, etc. However, programmers are reluctant to write API protocols. Even when available, there is no guarantee of their consistence, completeness, and correctness. Dynamic Protocol Mining (DPM) [2]-[5] is a promising approach to infer useful API protocols automatically. It always works in a two-phase mode: 1) Program Execution Traces (PETs) are gathered from client programs leveraging instrumentation techniques. 2) API protocols are synthesized from PETs based on various kinds of sequential data mining techniques. Compared with Static Protocol Mining techniques (SPM) [6]-[13], DPM can achieve more accurate results based on runtime information. Additionally, it can be used more extensively, especially when source codes are unavailable. Furthermore, many tricky problems with SPM, such as infeasible paths, complicated data structures and pointer aliasing can be avoided. However, the following drawbacks limit its applications in industrial practice: 1) The effect of DPM largely depends upon input test cases. If an improper set of test cases is selected, DPM may neglect many program paths and cause partial and inaccurate API protocols. 2) In order to gather PETs, DPM is required to run client programs, which may be difficult to be automated in some cases. 3) The runtime overhead caused by instrumentation techniques may discount the practical usability of DPM. 4) Existing DPM tools (such as ADABU [14]) always work in a manner as follows. First, they collect PETs from client programs using a tracer and then store the traces into a trace file. Then, they take the trace file as input and synthesize API protocols. Each run of a client program will generate a trace file and corresponding protocols. Results of multiple runs cannot be merged. What is worse is that mined protocols are biased to input trace files. Additionally, if PETs gathered from a program is scarce, we may achieve partial and inaccurate API protocols. Even though mined protocols can be merged, to gather enough PETs for mining, we should manually run a large number of programs one by one, which will result in significant manpower overhead and is unacceptable in practice.

This paper aims to improve the DPM techniques and promote their applications in industrial practice. We present a DPM framework NSpecMiner, which is based on a client-server architecture. The client is a tracer, which collects PETs and sends them to the server for mining. The server receives PETs and synthesizes API protocols. After that, the mined protocols are stored on the server to provide various kinds of remote services, such as API protocol retrieval and program verification, etc. The functioning of our framework is illustrated in Figure 1.

(DOI reference number: 10.18293/SEKE2017-089)

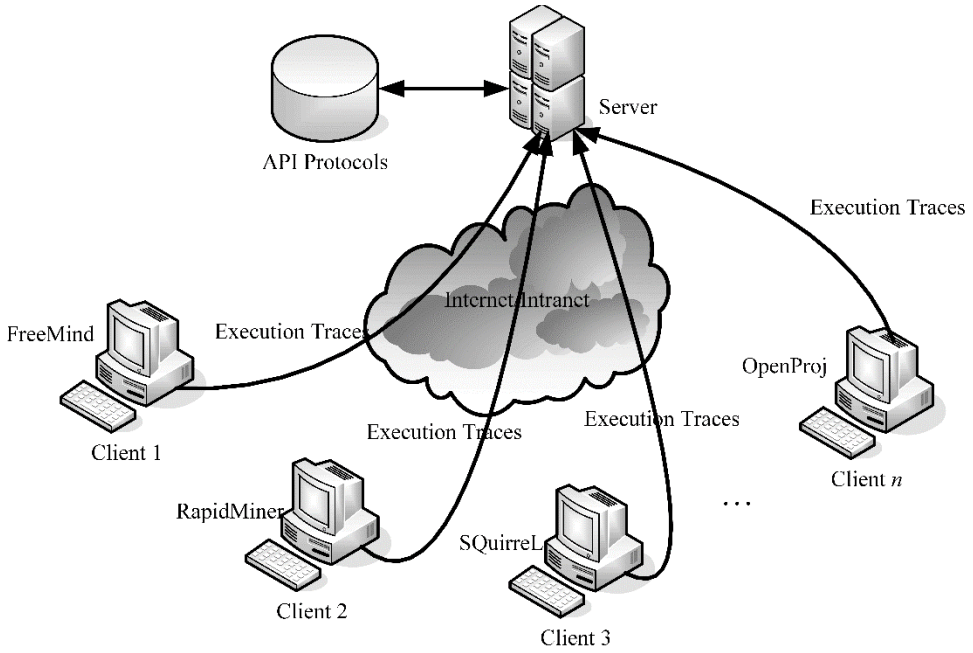


Figure 1 Functioning of NSpecMiner

Compared with local miners (such as ADABU), this architecture can benefit protocol mining in many ways: 1) Since the server can receive PETs from different clients and programs (as shown in Figure 1, the server receives PETs from *n* clients, which run different programs respectively, such as FreeMind, RapidMiner, SQuirreL and OpenProj), a large number of diverse PETs are likely to be collected efficiently, which is essential for mining accurate and complete API protocols. 2) The instrumentation overhead can be balanced among multiple clients based on a divide-and-conquer strategy. 3) This architecture is flexible, because client tracers based on whatever instrumentation techniques can send PETs to the server for mining if their formats satisfy requirements. 4) Via integrating the client tracer into daily used software, the task of mining protocols can be completely automated. On the client side, once the software is run by an end user, PETs will be sent to the server for mining automatically in the background. The whole mining process is transparent to software users. Little extra manpower is required to run client programs exclusively for gathering PETs. On the server side, thousands of millions of PETs may be received from different clients per second and perfect API protocols will be learned, which may cost several months by local miners.

The contributions of this paper are:

- A dynamic API protocol mining framework NSpecMiner based on a client-server architecture is proposed.
- A strategy of balancing instrumentation overhead is proposed.
- Experiments are conducted to evaluate our technique.

The rest of this paper is organized as follows: Section 2 presents the overview of our framework. Section 3 introduces an example API protocol mining technique used in this work. Section 4 elaborates our strategy of instrumentation balance. Section 5 presents our strategy of API protocol evolvement. Section 6 evaluates our technique and demonstrates preliminary results. Section 7 presents our conclusions and future work.

II. Overview of NSpecMiner

NSpecMiner is a generic dynamic API protocol mining framework. The distinguishing characteristic of it is that it is based on a client-server architecture, where clients collect PETs and send them to the server for mining. With the help of this framework, we can mine API protocols from application programs dynamically with little manpower overhead. Additionally, accurate and complete API protocols may be achieved.

The overview of NSpecMiner is illustrated in Figure 2, which mainly consists of a client and server. The client of NSpecMiner comprises a Proram Tracer and a Network Communication Module (NCM). The Program Tracer collects PETs from application programs and passes them to NCM. It gathers PETs based on instrumentation techniques, which insert binary codes into interested method bodies. Once an instrumented program is ran with test cases generated automatically or manually, the embedded codes will output information of method calls sequentially. Whatever instrumentation techniques and tools can be used in our framework (such as ASM [15], BCEL [16], Java agent [17] and Javassist [18]-[19]), provided that the format of output PETs can satisfy the requirement of NSpecMiner. After that, the client sends gathered PETs to the server via NCM, one method call after another.

The server receives method calls sequentially through NCM from multiple clients and passes them to the module of API Protocol Mining (APM). The APM is a key module for our

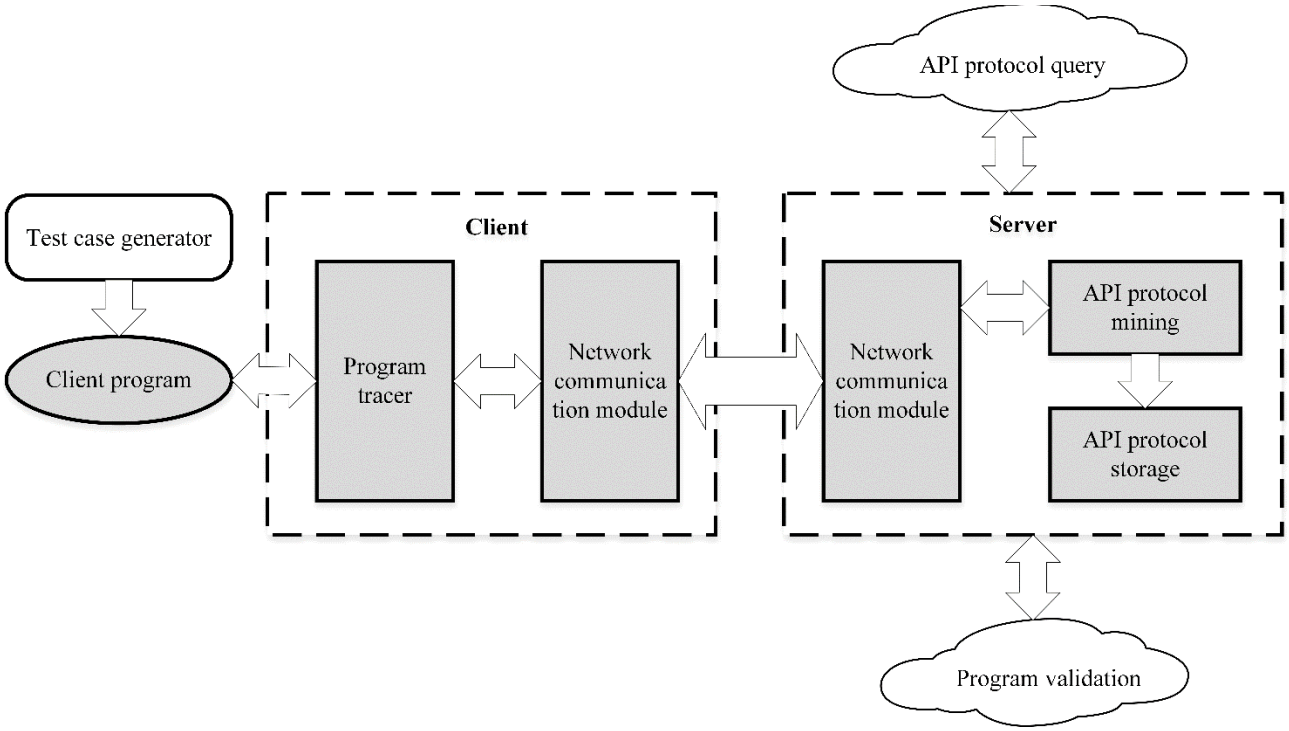


Figure 2 Overview of NSpecMiner

framework. It takes PETs as input and synthesizes API protocols based on sequential data mining techniques. For instance, [14], [20]-[23] mine API protocols based on Finite State Automaton (FSA). [24] models temporal specifications among Application Programming Interfaces (API) or Abstract Data Types (ADT) using Probabilistic Finite State Automaton (PFSA). Actually, whatever mining techniques based on sequential data can be used in our framework. In this paper, we utilize an online mining approach based on Markov model as an example to demonstrate the working principle of our framework. After that, NSpecMiner saves mined protocols on the sever through the module of API Protocol Storage (APS). Based on the protocols, our framework can provide various kinds of remote services, such as API protocol retrieval, program verification, etc.

NSpecMiner uses the TCP for communication and UDP is an inadvisable choice, because only the TCP can provide a reliable delivery service. Based on the TCP, all method calls sent sequentially to the network can be received in correct order on the server side, which is crucial for API protocol mining.

Via integrating the client tracer of NSpecMiner into widely used software, PETs will be sent to the server for mining automatically. The whole process is undergone in the background transparently. On the other hand, since the server can receive PETs from multiple clients (or programs), it is likely to achieve a large number of diverse PETs, which is essential for mining accurate and complete API protocols.

III. MINING API PROTOCOL

NSpecMiner is a generic framework, which can utilize various kinds of protocol mining techniques. In this paper, we use the online mining approach proposed by Chen et al. [25] as an example to demonstrate the working principle of our framework.

Chen et al. [25] mined API protocols based on an extended Markov model with final probability (MCF). The formal definition of MCF is given below:

Definition 1 (Markov chain with final probability). A *Markov Chain with Final Probability* (*MCF*) M is a 4-tuple (Q, τ, π, γ), where Q is a set of states, τ: $Q \times Q \rightarrow [0, 1]$ is the transition probability function, which is always described using a transition matrix P, π: $Q \rightarrow [0, 1]$ is the probability distribution over initial states, γ: $Q \rightarrow [0, 1]$ is the probability distribution over final states. The functions π and γ must satisfy the requirements: $\sum_{q \in Q} \pi(q) = 1$ and $\sum_{q \in Q} \gamma(q) = 1$.

Compared with traditional Markov model, MCF has an additional probability distribution over final states (Final Probability), which indicates how probable a chance process will end with a state.

Relying on the MCF, Chen et al. modeled API protocols by regarding states as methods and transitions as temporal relationships among methods. Figure 3 shows an example API protocol of Java class java.io.FileOutputStream described

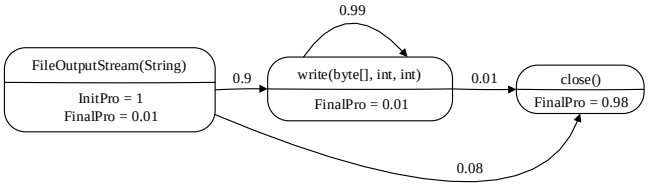


Figure 3 API protocol of Java class FileOutputStream described using MCF.

using the MCF. As we can see, the rounded rectangles are states, which are labeled with method signatures above a line. Arrows denote transitions, which are labeled with transition probabilities. InitPro and FinalPro are initial probability and final probability, respectively. What should be noted is that all states have properties InitPro and FinalPro. We omit the ones whose value is zero.

Given a repository of PETs *R*, Chen et al. learned MCFs from *R* using an online approach. In detail, they received a method call of each PET sequentially. For each method call, they evolved the corresponding MCF until it was good enough. The online approach has minimum space overhead. Additionally, since it can evolve API protocols persistently, accurate and complete protocols may be achieved.

Finally, they transformed a MCF to a Nondeterministic Finite-state Automaton (NFA) by discarding infrequent transitions and probabilities. The final API protocols described using NFA can be used for program verification, testing, documentation, etc.

IV. Instrumentation Balance

The time overhead incurred by instrumentation may cause performance issue to software running on the client machine and discount the practical usability of our framework. To mitigate the problem, we propose to balance instrumentation overhead among multiple clients.

Let's assume that we aim to mine API protocols of a set of classes *U*. It may cause much runtime overhead if we instrument all the classes in a single application program. To avoid this situation, we distribute the instrumentation task to multiple client programs via the instrumentation set. Let A be a set of application programs required to be instrumented. Given a program $a \in A$, the instrumentation set of a from U denoted by $\mathrm{IMTD}(a, U)$ is a subset of U, i.e., $\mathrm{IMTD}(a, U) \subseteq U$. The public methods of a class c will be instrumented in a run of program a, only if $c \in \mathrm{IMTD}(a, U)$. Obviously, via assigning program a a reasonable instrumentation set, we can confine its instrumentation overhead to an acceptable range. We perform this task based on the following method. First, we compute the size of the instrumentation set based on a function $\varphi: A \longrightarrow N$, which is given below:

$$\varphi(a) = \begin{cases} \varphi_1 & a \text{ is a hard real} - \text{time progam} \\ \varphi_2 & a \text{ is a soft real} - \text{time program} \\ \varphi_3 & a \text{ is a non} - \text{time} - \text{critical program} \end{cases}$$

where φ_1, φ_2 and φ_3 are specified constant values and satisfy the requirement $\varphi_1 < \varphi_2 < \varphi_3$. Our consideration is that less classes should be instrumented for time-critical programs than non-time-critical programs. After that, we select classes from U and add them to $\mathrm{IMTD}(a, U)$. What should be noted is that classes in U may not be covered by program a. To address this problem, we select classes from $\mathrm{CMTD}(a) \cap U$, where $\mathrm{CMTD}(a)$ is the set of classes covered by program a. Additionally, the following cases are considered:

1. $\mathrm{CMTD}(a) \cap U = \emptyset$, let $\mathrm{IMTD}(a, U) = \emptyset$.
2. $\mathrm{CMTD}(a) \cap U \neq \emptyset \wedge |\mathrm{CMTD}(a) \cap U| \leq \varphi(a)$, let $\mathrm{IMTD}(a, U) = \mathrm{CMTD}(a) \cap U$.
3. $\mathrm{CMTD}(a) \cap U \neq \emptyset \wedge |\mathrm{CMTD}(a) \cap U| > \varphi(a)$, we select $\varphi(a)$ classes from $\mathrm{CMTD}(a) \cap U$ based on a class selection algorithm and add them to $\mathrm{IMTD}(a, U)$.

In words, if the number of classes included in $\mathrm{CMTD}(a) \cap U$ is less than $\varphi(a)$, we add all classes in $\mathrm{CMTD}(a) \cap U$ to $\mathrm{IMTD}(a, U)$. Otherwise, we select as many as $\varphi(a)$ classes from $\mathrm{CMTD}(a) \cap U$.

In order to collect enough PETs, local protocol miners may instrument all classes in U at one time, which may cause much runtime overhead. Relying on the strategy of instrumentation balance, we can collect as many PETs as local protocol miners with little runtime overhead, which improves the practical usability of our framework significantly. It should be noted that our instrumentation balance strategy is based on the granularity of classes rather than methods. The reason for this is that partial instrumentation of a class will cause imperfect PETs which may lead to inaccurate API protocols [26].

V. API Protocol Evolvement

The network load overhead incurred by our framework may increase the cost of mining API protocols. In this section, we introduce our strategy of API protocol evolvement, which can reduce the network load overhead to some extent.

It should be noted that, there is no necessity to evolve API protocols persistently. Once a protocol is good enough, we can stop refining it, which can reduce the network load overhead and save computational resources on both client and server sides. In order to accomplish the task, a metric used to measure the goodness of API protocols is required. Currently, there does not exist such a metric and measuring the goodness of API protocols accurately and automatically is challenging. In this work, we perform the task approximately based on the following heuristic: let c be a class, p be the API protocol of c, which is described using a MCF: (Q, τ, π, γ). If the following requirements (*Good-Enough Requirement*) are satisfied, we believe p is good enough.

- $Q \supseteq \mathrm{PM}(c)$, where $\mathrm{PM}(c)$ denotes the set of public methods of class c (a single-object API protocol subsumes only public methods of a class);
- p keeps $\delta - constant$ in the continuous evolvement for T_e times, where T_e is the specified *Evolving Threshold*. Let $p_i: (Q_i, \tau_i, \pi_i, \gamma_i)$ and $p_{i+1}: (Q_{i+1}, \tau_{i+1}, \pi_{i+1}, \gamma_{i+1})$ be the API protocols achieved before and after the *i*th evolvement of p respectively. We say p is $\delta - constant$ in this evolvement if it satisfies the following requirements: 1) $Q_i = Q_{i+1}$; 2) $|\tau_i(t) - \tau_{i+1}(t)| \leq \delta$, where t denotes a common transition of p_i and p_{i+1}; 3) $|\pi_i(q) - \pi_{i+1}(q)| \leq \delta$, where q is a common state of p_i and p_{i+1}; and 4) $|\gamma_i(q) - \gamma_{i+1}(q)| \leq \delta$.

In detail, for each API protocol p, we maintain two variables n_p and n_e. The former records the number of methods included in p. The latter denotes the count of continuous evolvement, in which p is $\delta - constant$. At the end of each evolvement, we update n_p. As to n_e, we compare the protocols achieved before and after each evolvement. If p is $\delta - constant$, we have $n_e \leftarrow n_e + 1$, otherwise $n_e \leftarrow 0$. Once $n_p \geq n_c$ and $n_e \geq T_e$ where

n_c is the number of public methods included in class c, we notify all clients to stop instrumenting class c. Since no more PETs regarding c will be received, the evolvement of p on the server side will stop automatically.

In some cases, we need to restart the evolvement of good-enough protocols. For example, let's assume that p is a good-enough protocol of class c. The API protocol of c may be changed in the upgraded version of c. Thus, we should restart the evolvement of p until the good-enough requirement is satisfied in terms of the latest version of c. This task can be accomplished by notifying clients to restart instrumenting class c.

VI. Preliminary Results

In order to investigate the feasibility of our technique, we implemented NSpecMiner based on a previous prototype tool ISpecMiner [25] and called the novel tool ISpecMiner+. ISpecMiner and ISpecMiner+ are nearly the same: 1) both tools employ the Java agent [17] technique to instrument client programs; 2) both tools mine API protocols based on the online approach proposed by Chen et al. [25]. The only difference is that ISpecMiner is a local miner, while ISpecMiner+ is a distributed API protocol miner.

To evaluate our approach, we performed a comparison test with ISpecMiner and ISpecMiner+. Our experiment proceeds as follows. We used ISpecMiner and ISpecMiner+ to mine API protocols from a same set of programs. After that, we investigated API protocols achieved by ISpecMiner and ISpecMiner+, respectively. The experimental setup was exactly the same as that of [27]: 1) the subject programs used for mining were four real-world Java programs FreeMind, RapidMiner, SQuirreL SQL Client and OpenProj; and 2) 10 JDK classes from java.io and java.util were instrumented and investigated. Detailed information about the experimental setup please refer to [27]. ISpecMiner was configured to run each subject programs once sequentially. For each program, ISpecMiner instrumented all 10 JDK classes. Since ISpecMiner mines API protocols using an online approach, results of each ran will be merged automatically. ISpecMiner+ worked in a LAN environment similar to that shown in Figure 1, which consists of a server and four clients. Information of the server and client computers is summarized in Table I. On the server, ISpecMiner+ was configured to receive PETs from port 5123. Each client computer ran a subject program and instrumented all 10 JDK classes. To avoid biases, a subject program was feed with the same test cases under ISpecMiner and ISpecMiner+.

By analyzing experimental results, we found that API protocols mined by ISpecMiner and ISpecMiner+ were exactly the same. It indicates that NSpecMiner is effective to mine useful API protocols as local miners. While our technique is able to gather PETs concurrently from multiple clients. For each client user, only one subject program is required to run. If we integrate the client tracer of NSpecMiner into daily used software, API protocols can be synthesized on the server automatically and transparently. Little manpower overhead is required to exclusively run client programs for gathering PETs.

TABLE I. INFORMATION OF NETWORK COMPUTERS

Type	Number	Information
Server	1	Intel(R) Core(TM) i7-3770k CPU 3.9GHz, 8GB of memory, running Windows 7
Client	2	Intel(R) Core(TM) i3-2100 CPU 3.1GHz, 3GB of memory, running Windows XP
Client	1	Intel(R) Core(TM)2 Duo CPU 2.93GHz, 2GB of memory, running Windows XP
Client	1	Intel(R) Core(TM)2 Duo CPU 2.93GHz, 2GB of memory, running Windows 7

VII. Conclusions and Future Work

In this paper, we proposed a generic DPM framework NSpecMiner. The most distinguishing characteristic of our framework is that it is based on a client-server architecture. The distributed technology has been used widely in many aspects of software development, such as the Cooperative Bug Isolation Project [28], which collects program runtime information for tracking down bugs using a method similar to ours. However, to the best of our knowledge, NSpecMiner is the first DPM framework based on the distributed technology. As elaborated in this work, the client-server architecture can mitigate many limitations of DPM techniques, such as the overfitting problem incurred by scarce PETs, the difficulty to be automated, much runtime overhead caused by instrumentation, etc. Compared with local dynamic miners, NSpecMiner is able to mine accurate and complete API protocols with much lower cost, which increases the practical usability of DPM techniques significantly. In the evaluation, we performed a comparison test with local API protocol miners and our framework. Preliminary results show that our framework is effective to mine useful API protocols as local miners. While NSpecMiner is able to gather PETs concurrently from multiple clients and other advantages of the distributed technology will further benefit DPM significantly. If we integrate the client tracer of NSpecMiner into daily used software, API protocols can be synthesized on the server automatically and transparently. Little manpower overhead is required.

In conclusion, the distributed technology is significantly valuable for DPM and may promote its applications in industrial practice. Although the distributed technology is a common approach, there exist many particular challenges while using this approach for DPM, such as the strategy of instrumentation balance, the evolvement of API protocols, etc., which deserve much more research effort. Although some unforeseen issues regarding scalability, privacy and security may be raised when using our framework in practice, we are confident that they can be resolved relying on today's available technology. Additionally, along with the progress of network technology, our framework will be more useful. In this work, we presented a general view of NSpecMiner and only a preliminary evaluation result was given. Detailed discussions and more extensive evaluations of our framework are left as an extension of this work.

Acknowledgment

This work was supported by the Youths Science Foundation of Wuhan Institute of Technology (No. k201622), Surveying and Mapping Geographic Information Public Welfare Scientific Research Special Industry (No. 201412014), Educational Commission of Hubei Province (Q20151504), National Natural

Science Foundation of China (No. 41501505, 61502355 and 61502354) and the Key Program of Higher Education Institutions of Henan Province (No. 17A520040).

REFERENCES

[1] M. Pradel, T.R. Gross, "Leveraging test generation and specification mining for automated bug detection without false positives," in Proceedings of the 34th International Conference on Software Engineering, Zurich, Switzerland, pp. 288-298, 2012.

[2] M. P. Robillard, E. Bodden, D. Kawrykow, et al., "Automated API property inference techniques," IEEE Transactions on Software Engineering, 2013, 39 (5): 613-637.

[3] M. Gabel, Z. Su, "Javert: fully automatic mining of general temporal properties from dynamic traces," in Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering, ACM, Atlanta, 2008.

[4] D. Chen, Y. Zhang, R. Wang, et al., "Extracting more object usage scenarios for API protocol mining," Proceedings of the 27th International Conference on Software Engineering and Knowledge Engineering, 2015: 607-612.

[5] D. Chen, Y. Zhang, R. Wang, et al., "Mining API protocols based on a balanced probabilistic model," Proceedings of the 12th International Conference on Fuzzy Systems and Knowledge Discovery, 2015: 2276 - 2282.

[6] M.K. Ramanathan, A. Grama, S. Jagannathan, "Static specification inference using predicate mining," SIGPLAN Not., vol. 42, pp. 123-134, 2007.

[7] S. Shoham, Y. Eran, S. Fink, et al., "Static specification mining using automata-based abstractions," in Proceedings of the 2007 International Symposium on Software Testing and Analysis, ACM, London, 2007.

[8] M. Di Penta, L. Cerulo, L. Aversano, "The life and death of statically detected vulnerabilities: An Empirical Study," Information and Software Technology, vol. 51, pp. 1469-1484, 2009.

[9] S. Thummalapenta, T. Xie, "Alattin: mining alternative patterns for defect detection," Automated Software Engineering, vol. 18, pp. 293-323, 2011.

[10] D. Lo, G. Ramalingam, V.P. Ranganath, et al., "Mining quantified temporal rules: formalism, algorithms, and evaluation," Science of Computer Programming, vol. 77, pp. 743-759, 2012.

[11] Z. Li, Y. Zhou, "PR-Miner: automatically extracting implicit programming rules and detecting violations in large software code," SIGSOFT Softw. Eng. Notes, vol. 30, pp. 306-315, 2005.

[12] M.K. Ramanathan, A. Grama, S. Jagannathan, "Path-sensitive inference of function precedence protocols," in Proceedings of the 29th International Conference on Software Engineering, IEEE Computer Society, 2007.

[13] M. Acharya, X. Tao, X. Jun, "Mining interface specifications for generating checkable robustness properties," in Proceedings of the 17th International Symposium on Software Reliability Engineering, 2006.

[14] V. Dallmeier, C. Lindig, A. Wasylkowski, et al., "Mining object behavior with ADABU," in Proceedings of the 2006 International Workshop on Dynamic Systems Analysis, ACM, Shanghai, 2006.

[15] ASM, http://asm.ow2.org, 2016.

[16] BCEL, http://commons.apache.org/proper/commons-bcel, 2016.

[17] P. Caserta, O. Zendra, "JBInsTrace: a tracer of Java and JRE classes at basic-block granularity by dynamically instrumenting bytecode," Science of Computer Programming, vol. 79, pp. 116-125, 2014.

[18] S. Chiba, M. Nishizawa, "An easy-to-use toolkit for efficient Java bytecode translators," in Proceedings of the 2nd International Conference on Generative Programming and Component Engineering," Springer-Verlag, New York, 2003.

[19] M. Tatsubori, T. Sasaki, S. Chiba, et al., "A bytecode translator for distributed execution of "legacy" Java software," in Proceedings of the 15th European Conference on Object-Oriented Programming, Springer-Verlag, 2001.

[20] A. Wasylkowski, "Mining object usage models," in Companion to the Proceedings of the 29th International Conference on Software Engineering, IEEE Computer Society, 2007.

[21] D. Lorenzoli, L. Mariani, M. Pezz, "Automatic generation of software behavioral models," in Proceedings of the 30th International Conference on Software Engineering, ACM, Leipzig, 2008.

[22] R. Alur, P. Cerny, P. Madhusudan, et al., "Synthesis of interface specifications for Java classes," SIGPLAN Not., vol. 40, pp. 98-109, 2005.

[23] L.Mariani, F. Pastore, M. Pezze, "Dynamic Analysis for Diagnosing Integration Faults," IEEE Transactions on Software Engineering, 2011, 37(4): 486-508.

[24] G. Ammons, R. Bodik, J.R. Larus, "Mining specifications," SIGPLAN Not., vol. 37, pp. 4-16, 2002.

[25] D. Chen, R. Huang, B. Qu, et al., "Ming class temporal specification dynamically based on extended Markov model," International Journal of Software Engineering and Knowledge Engineering, 2015, 25(3): 573-604.

[26] J. Yang, D. Evans, D. Bhardwaj, et al., "Perracotta: mining temporal API rules from imperfect traces," in Proceedings of the 28th International Conference on Software Engineering, ACM, Shanghai, 2006.

[27] D. Chen, Y. Zhang, R. Wang, et al., "Mining universal specification based on probabilistic model," Proceedings of the 27th International Conference on Software Engineering and Knowledge Engineering, Pittsburgh, PA, USA, 2015: 471-476.

[28] The Cooperative Bug Isolation Project, http://research.cs.wisc.edu/cbi/, 2016.

Automatic Type Inference for Proactive Misconfiguration Prevention

Xiangyang Xu, Shanshan Li, Yong Guo, Wei Dong, Wang Li, Xiangke Liao
College of Computer Science
National University of Defense Technology
Changsha, China
Email: {xuxiangyang11,shanshanli,yguo,wdong,liwang2015,xkliao} @nudt.edu.cn

***Abstract*—Misconfigurations have become a major cause of software failures. Most research focuses on misconfiguration diagnosis and troubleshooting, which occur after the misconfigurations have happened. Actually, if we can prevent misconfiguration before software runs, many potential catastrophic failures of systems can be avoided, thus reducing customers' downtime and support costs. In software configuration, we found that most configuration options have specific constraints, which have a strong connection with the configuration option type. If we can check the configuration settings against the inferred type before the software runs, many misconfigurations can be prevented. In this paper, we explore a name-based method called ConfTypeInferer to automatically infer the type of configuration options, which can help users to correctly configure and check settings, thus preventing misconfigurations proactively. We manually studied several popular open-source software projects to investigate the classification and naming conventions of configuration option. Based on these findings, we designed and implemented the ConfTypeInferer. We performed comprehensive experiments to evaluate the effectiveness of our method.**

***Index Terms*—misconfiguration prevention; configuration option type inference; name-based analysis;**

I. Introduction

In recent years, configuration issues have drawn tremendous attention for their increasing prevalence and severity [1][2]. One important reason for today's prevalent configuration issues is the ever-increasing complexity of configuration, which is reflected by the large and still increasing number of configuration options. For example, hundreds of configuration options need to be set for the running of database servers [3] and web servers [4]. The runtime environment of software can be determined from the settings of the configuration options, such as memory size, time interval, resource limits, etc. To set them correctly, users need to understand the meaning of each configuration option. Unfortunately, although most software manuals contain a detailed description of how to set each configuration option correctly, it is non-trivial for users to find out what the corresponding guidance is. Users rarely have the patience to look at thousands of pages of documents. They always make configurations based on experience, or exaggerate a bit, by intuition, which leads to many misconfiguration issues.

Faced with these issues, many previous studies [5] [6] [7] [8] have devoted their work to misconfiguration diagnosis and troubleshooting, which occurs after the misconfigurations have happened. Actually, if we can prevent or detect misconfigurations before software runs, many potential catastrophic failures of systems can be avoided, thus reducing not only the customers' downtime but also the support costs. Some researchers have noticed that and already conducted some work to explore misconfiguration prevention. Xu et al. [9] check parameter settings at the system's initialization time to reduce damage from failures. Encore [10] detects software misconfigurations by exploiting the interaction between the configuration settings and the executing environment, as well as the rich correlations between configuration entries. Actually, we found that most configuration options have specific constraints, which have a strong connection to the configuration option type. Fig. 1 gives two real-world examples of misconfigurations caused by an invalid type. Without knowing the option type, a user's settings often violate configuration constraints, which results in many misconfigurations. These misconfigurations waste a tremendous amount of user effort for diagnosis and might cause severe failure in the system's runtime. If we can check these configuration settings against the inferred type before the software runs, many misconfigurations can be prevented. In this paper, we try to prevent misconfiguration proactively by inferring the configuration option type for users, which can be used for checking the configuration settings, thus avoiding

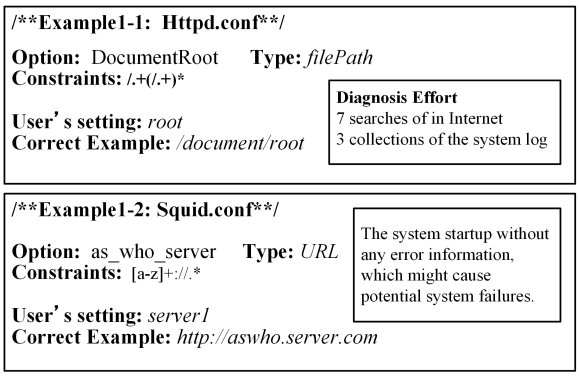


Fig. 1. Example of misconfiguration caused by invalid type. In example 1-1, the user takes a *path* type for a *filename* type, and the diagnosis cost lots of user effort. In example 1-2, the user sets a URL configuration option as a server name, which is ignored by system.

DOI reference number:10.18293/SEKE2017-072

many potential misconfigurations. We explore a name-based method called ConfTypeInferer to automatically infer the type of the configuration option.

Some challenges need to be addressed. First, we need a specific and comprehensive classification for configuration options, which is the basis of the type inference. Second, we need to figure out the naming conventions for the configuration options, which is the basis of semantic extraction. Third, we need to verify whether the configuration-option name really contains enough semantic information for type inference, or, in other words, whether the name-based method works.

To address these challenges, we manually analyzed several popular open-source software projects, such as PostgreSQL, Httpd, Nginx, Squid, etc. After a thorough study of the classification and naming conventions for their configuration-option names, we investigated the feasibility of the name-based method. Except for the *enumeration* type, the method works well for inferring most configuration option types. Then we designed and implemented ConfTypeInferer by combining name-based analysis with program analysis, in which the program analysis is to make up for the deficiency of the name-based analysis in inferring the *enumeration* type of configuration option and to verify the type inferred.

Our contributions are summarized as follows:

- Through manual analysis of several popular open-source software projects, we summarized several instances of finding configuration options, verifying the feasibility of the name-based method (Section II).
- We designed and implemented the architecture for ConfTypeInferer, the name-based method used to infer the type of configuration option (Section III and Section IV).
- We conducted comprehensive experiments to evaluate the effectiveness of ConfTypeInferer. Our results show that the accuracy of type inference can reach over 90%, and at the same time, it can prevent many misconfigurations (Section V).

II. The Feasibility of the Name-Based Method

To evaluate the feasibility of the name-based method, we try to answer the following research questions:

RQ1: How many and what types of configuration option exist in open-source software projects? Answering this question will provide a classification for type inference.

RQ2: What are the naming conventions for configuration options in open-source software projects? Answering this question will give us an in-depth understanding of configuration option names, and provide us with some ideas for mining semantic information from configuration option names.

RQ3: Do these configuration-option names convey enough information for type inference? Answering this question is the key to verify the feasibility of the name-based method.

To answer the above questions, we empirically studied more than 1,000 configuration options in several popular open-source software projects. The main findings of our study are illustrated as follows.

TABLE I
Coverage of Our Classification in Open-Source Software.

Software	Number	Coverage(%)
Redis-3.2.3	70	98.6
PostgreSQL-9.6rc1	269	98.1
Lighthttpd	273	96.2
Postfix-2.5	109	94.7
Squid-3.5.21	340	90.0
MySQL-5.7.15	732	87.7
Httpd-2.4.23	634	86.1
Nginx-1.10.2	637	85.1

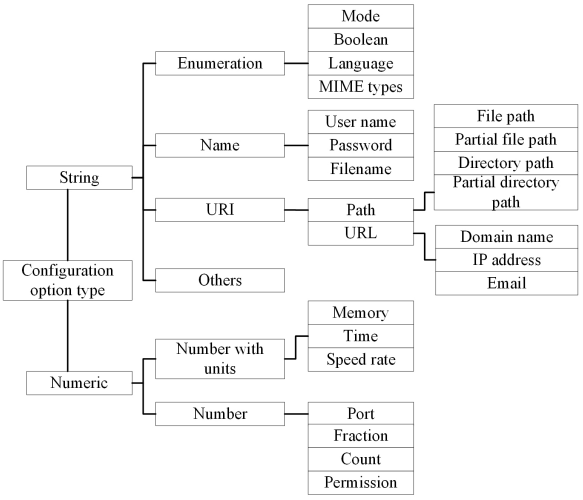


Fig. 2. Classification tree of configuration options

For RQ1, although some previous work [11] includes studies on type taxonomy, our aim is to generate configuration constraints by type inference. Therefore, we need a sufficiently fine-grained classification for all configuration options. We manually analyzed software manuals, configuration files, and even source code to classify each option. Fig. 2 illustrates our classification in the form of a tree. This classification tree can be supplemented with more software to be considered. We evaluated our classification effectiveness on about 3,000 options of eight open-source software systems by checking whether each option can be classified as one of types listed Fig. 2. As shown in Table I, the coverage of the classification is as high as 90% on average, with a minimum of 85.1%. Therefore, the result indicates the validity and efficiency of our classification.

For RQ2, we find that the configuration option names chosen by programmers are usually made up of readable words or common abbreviations connected by some separators. For example, in PostgreSQL, programmers use underscores to connect several words for the name of a configuration option, e.g., "listen_addresses," while in Httpd programmers use camel-case naming, e.g., "MaxRequestsPerChild," This naming convention makes it easier for us to perform text processing and extract semantic information. Besides, those words contained in configuration-option names usually convey explicit semantic information, including explanation, description, or constraints

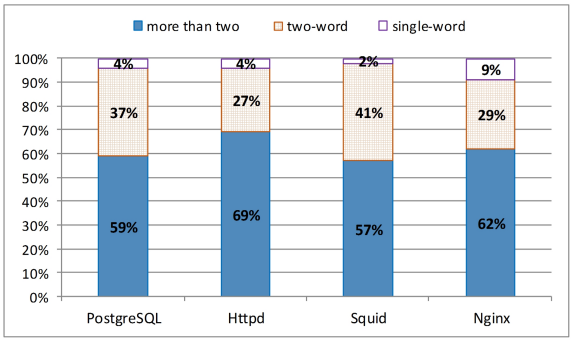


Fig. 3. The number of words in the configuration-option name.

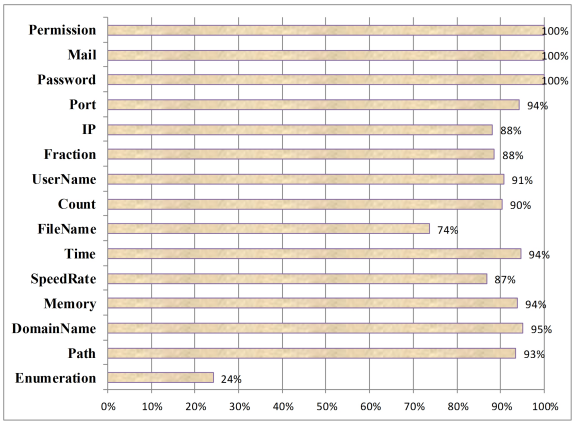


Fig. 4. The inference result of keyword matching.

about the configuration option, which reflect the option's type to some extent. For instance, the option names of *path* type usually contain keywords such as "directory," "location," "path," etc, and the option names of *memory* type usually contain keywords such as "memory," "buffer," "size," etc. These keywords can be used to build a dictionary for each configuration option type for keyword matching.

For RQ3, we studied the number of words in each configuration-option name. As Fig. 3 shows, the majority (91%-98%) of those configuration-option names contain more than one words, this observation enhances the feasibility of the name-based method because it is widely accepted that more words convey more information. In addition, we find that this name-based method doesn't work well in some cases. On the basis of the findings for RQ2, we established a dictionary for each type by collecting those high-frequency keywords in the configuration-option names, and implemented a simple program to infer the configuration option type through text processing and keyword matching. The inferring results are presented in Fig.4. We observe that the inference results are ideal (74%-100%) for most configuration option types. Unfortunately, the tool behaved poorly (only 24%) when inferring the *enumeration* type, whose ratio is relatively higher than the other type. An *enumeration* type option means the programmer enumerates all possible values to be set, while it's not impossible, but is difficult to infer the *enumeration* type from several words since there are so many words that can be used to express *enumeration* type. We chose abstract syntax tree (AST) analysis to address this problem (introduced in Section IV).

III. Architecture of ConfTypeInferer

Based on the findings in Section II, we designed and implemented ConfTypeInferer, an automatic type inferer for configuration options. Fig. 5 illustrates its architecture. In the extraction phase, we take configuration files as input, which could be the original template configuration files or user-specified ones. After parsing the configuration files through some parsing tool, we can determine the name of partial configuration options. In the mapping phase, we use the tool ConfMapper [12], which was designed to implement automatic mapping from configuration options to the relevant program variables, and we get pairs of option and variable. In the inferring phase, ConfTypeInferer conducts name-based analysis on the configuration-option names to infer the type. In addition, we identified the options for *enumeration* type by AST analysis. In the verifying phase, we conducted data-flow analysis for the program variable of each option to verify the type inferred previously.

We implemented the AST analysis and data-flow analysis on the basis of Clang [13], a C language family frontend for LLVM [14].

IV. Design and Implementation

This section presents detailed descriptions of the design and implementation of ConfTypeInferer. The goal of ConfTypeInferer is to design a user-oriented tool that automatically infers the configuration option type without any manual effort.

A. Extraction and Mapping

To infer the configuration option type, we need to extract the configuration options and corresponding variables, which is the basis of name-based analysis and program analysis. This question has been studied in our previous work [12]. We proposed a tool named ConfMapper to accomplish the automated mapping from the configuration options to the relevant program variables without needing to understand the complicated semantic context in the source code.

B. Inferring by Name-based Analysis

As discussed in Section II, the naming convention of most configuration options makes it possible for us to perform text processing and extract semantic information. However, many difficulties still exist in the implementation of name-based analysis. For instance, a keyword might reflect different configuration option types, and a name might contain several keywords that reflect different types, so how can we use the semantic information to infer the type in these cases? In ConfTypeInferer, we have come up with a scoring model to choose the most likely type. Specifically, we divide the name-based analysis into three steps. First, we separate the

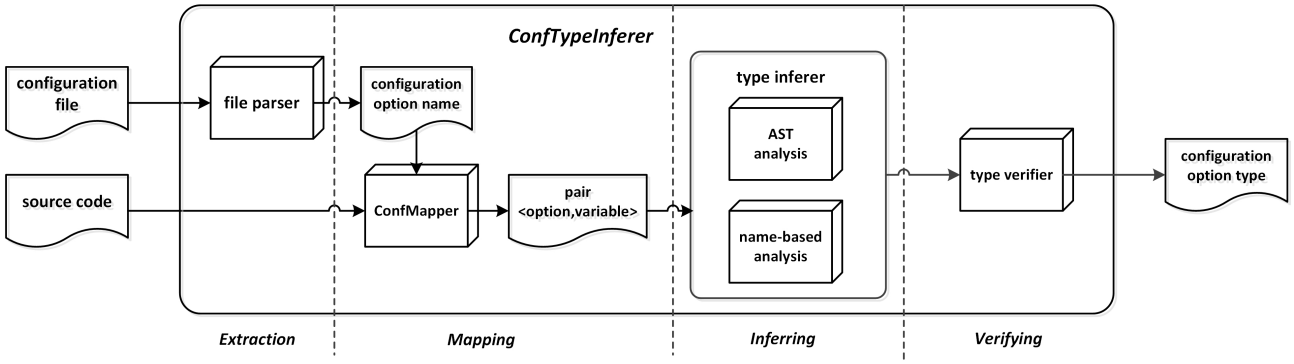


Fig. 5. The architecture of ConfTypeInferer.

configuration-option names into words by text processing, and establish a dictionary for each configuration option type, with the words ranked according to their frequency. Second, we use the scoring model to infer the configuration option type. Third, infer the *enumeration* type using AST analysis.

1) Word Segmentation and Dictionary Establishment: Based on the naming convention for the configuration option, we use uppercase letters, underscore ("_"), dot (“.”), and hyphen ("−") as separators to get words contained in the configuration-option name. We build a dictionary for each configuration option type by categorizing the words according to different types. In each dictionary, these words are ranked by their frequency of occurrence.

2) Score Model: As a name might contain several keywords, which reflect different types, and a keyword might reflect different configuration option type, the score model is described in (1).

$$score_{type} = \sum_{word_i}^{n} score_{freq} \cdot score_{order}\ 1 \le i \le n \qquad (1)$$

Here *n* means the number of keywords in an option name, $word_i$ represents the *i*th keyword, and $score_{type}$, $score_{freq}$, and $score_{order}$ mean the total score for a certain configuration option type, the frequency score for a matched word, and the order score for a matched word, respectively. The type with highest score is the inference result.

More concretely, $score_{freq}$ is set in the step for dictionary establishment, according to the word's frequency of occurrence. In our tool, we set three score levels using the function described in (2). The *x* is the frequency rank of a word.

$$score_{freq} = \begin{cases} 3 & 1 \le x \le 5 \\ 2 & 6 \le x \le 10 \\ 1 & x \ge 11 \end{cases} \qquad (2)$$

The $score_{order}$ is set according to the word's position in a configuration option name. We find that those words at the front of a configuration option name are usually used to describe the words following them, which means that the last word is most likely to reflect its type, e.g., “listen_addresses” “ssl_cert_file,” and “temp_buffers.” This is a common habit when people are naming a thing. However, there are exceptions, such as if the programmer has a strange naming habit, which would affect our inference result. In our tool, we set two score levels and, usually, the last word of a name has a higher score.

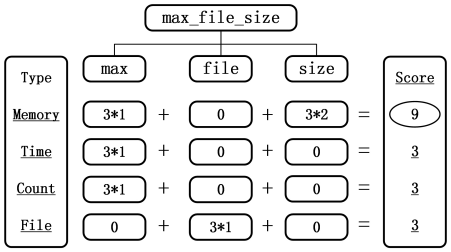


Fig. 6. An example of calculation of $score_{type}$

Fig. 6 gives an example of the calculation of $score_{type}$. The configuration-option name “max_file_size” contains three keywords that reflect different types, among which the keyword “max” could reflect further different types including memory, time, and count. After doing the calculation shown in Fig. 6, we can infer the correct type.

C. Inferring Enumeration Type

As mentioned in Section II, it is not enough to infer the configuration option type only by name-based analysis. We tried to use the program analysis method to address this issue. In order to infer the options for *enumeration* type, we manually analyzed their program variables in the source code. We find that programmers assign values to these variables by a similar pattern. In general, most software does this assignment by specific structures or functions whose name contains the configuration option or its variable, and the body of structures or functions contains many macros and strings related to the configuration option. This assignment pattern can be located and identified by AST analysis. In addition, we can also obtain all possible values of an *enumeration* type option by AST analysis.

D. Verifying the Inferred Type

We verify options' types by checking their variables' data type. This step is mainly based on our finding relating to the close connection between the configuration option type and the relevant program variables data type. For example, for a *path* type option, its variable must a string, while a *memory* type might have a variable of integer or float-point number type. In the mapping phase, we have determined the pairs of option and variable, and the data type of the variable can be obtained by a data-flow analysis.

E. Constraints Enhancement

We envisage that ConfTypeInferer is the first step towards a generic and systematic solution to prevent configuration errors. With the option type inferred, we can define constraints for a specific type, which can be used for misconfiguration checking in source code and configuration files.

1) Comments Enhancement in Configuration files: Commenting configuration files is a common practice in software development; the comments are direct, descriptive, and easy-to-understand, which can give users guidance on configuring correctly. However, we have found that the amount of useful information contained in comments is very limited in current software. With the constraints for specific option types, we can enhance comments and provide clues for user's configuration.

2) Misconfiguration Checking in Source Code: One reason for the difficulty in troubleshooting misconfigurations is the lack of diagnostic information. We can use the option type inferred to enhance the configuration constraints in the source code, which can be used for misconfiguration checking, thus improving the system's reliability and preventing many potential configuration errors.

V. Evaluation

In this section, we discuss the comprehensive experiments conducted to evaluate the effectiveness of ConfTypeInferer on two aspects: the accuracy of type inference and the effectiveness of misconfiguration prevention.

A. Accuracy of Type Inference

As we completed our study, we chose eight popular open-source software projects to evaluate the accuracy of type inference. Their information is shown in Table II. Note that there are four software projects(MySQL, Redis, Lighthttpd, and Postfix) that are not used for building dictionaries, and we chose them to verify that our name-based method works well on other C/C++ software projects. We get all the options and their correct types by manually viewing the software documentation, configuration files, and even the source code.

As Fig. 7 shows, the accuracy of type inference in open-source software projects can reach over 90%. This is an acceptable result considering the large number of configuration options. The results for MySQL, Redis, Lighthttpd, and Postfix verify that this method works well on other C/C++ software projects.

TABLE II
List of Software for Experiments

Software	Description	Software	Description
PostgreSQL	Database	Nginx	Reverse proxy
Lighthttpd	Web server	Squid	Web delivery
MySQL	Database	Redis	Database
Httpd	Web server	Postix	Mail proxy server

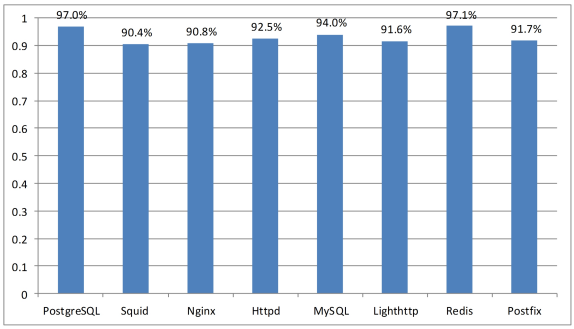


Fig. 7. Accuracy of type inference in open-source software. The percentages in the figure represent the correct inference percentages for the configuration options.

B. Effectiveness of Misconfiguration Prevention

To evaluate ConfTypeInferer's effectiveness in misconfiguraiton prevention, we performed two experiments with four software projects: PostgreSQL, Httpd, Nginx, and Postfix. In the first experiment, we injected random errors into correctly configured systems and used ConfTypeInferer to detect the injected errors. In the second experiment, we applied ConfTypeInferer to check against real-world misconfiguration problems.

1) Injected Misconfigurations: For each software project, we randomly injected 20 errors with SPEX-INJ [15] into the configuration files. SPEX-INJ automatically generates configuration errors by violating the constraints. As Table III shows, we detected most constraint violations using the option's inferred type.

2) Real-world Misconfigurations: We searched forums, frequently asked questions (FAQs) pages, and configuration documents to find actual configuration problems that users have experienced with our target software projects. In total, we chose eight representative misconfigurations to reproduce. These misconfigurations are type-related and caused by errors in the configuration files. We tried to detect these misconfigurations, given the option type inferred. Table IV lists the configuration errors for each software projects, as well as the detection

TABLE III
The number of injected misconfigurations detected by ConfTypeInferer

Software	Total	Detected
PostgreSQL	20	16
Httpd	20	13
Nginx	20	12
Postfix	20	15

TABLE IV
DETECTION OF REAL-WORLD MISCONFIGURATIONS

ID	Software	Problem Description	Success
1	PostgreSQL	Logging is not performed because log_directory (path) is set incorrectly	Y
2	PostgreSQL	Query operation is very slow due do the work_mem (memory) option being set too low	N
3	Httpd	Website visitors are unable to upload files due to the wrong permission (permission) being set	N
4	Httpd	Unable resolve PHP code due to setting the AddType (enumeration) option as a freedom string	Y
5	Nginx	File creation error due to datadir's wrong owner (username)	Y
6	Nginx	Failed to connect to the proxy server due to the wrong proxy_pass (url) being set	Y
7	Postfix	Cannot deliver mail locally due to the mydestination (email) option being set incorrectly	Y
8	Postfix	Cannot forward user's email to the Internet due to the relayhost (email) option is set incorrectly	Y

results. Table IV shows many (6/8) misconfigurations can be prevented by type checking. However, some misconfigurations cannot be prevented even when the type is inferred. For example, for Problem ID 3, you need more detailed constraints to set a *memory* option. This limit has inspired our future work for inferring more detailed constraints based on the inferred type.

VI. RELATED WORK

To prevent misconfigurations, some research detects misconfigurations by constraint verification. ECC Fixer [16] infers configuration constraints by program analysis and detects the configuration violations, it designs an algorithm that automatically generates range fixes for a violated constraint. SPEX [15] infers configuration constraints from source codes, and use these constraints to harden systems against configuration errors and to detect error-prone designs. These methods all neglect the semantic information of configuration-option names, which can complement program analysis.

The identifier names chosen by developers convey information about the semantics of a program. This information can complement traditional program analyses in various software engineering tasks, such as bug finding, code completion, and documentation. Recent work uses identifier names to infer API specifications [17], to synthesize code completions [18], and to detect incorrectly ordered method arguments of the same type [19].

VII. CONCLUSION

Misconfigurations have become a major cause of software failures. In this paper, we have explored a name-based method called ConfTypeInferer to automatically infer the type of a configuration option, which can help users to configure correctly and check their settings, avoiding many unnecessary misconfigurations. We manually studied several popular open-source software projects and research on the classification and naming conventions for configuration option. Based on these findings, we designed and implemented the ConfTypeInferer. We conducted comprehensive experiments to evaluate the effectiveness of ConfTypeInferer.

ACKNOWLEDGMENT

This paper is partially supported by NSFC No. 61532007, No. 61690203, and No. 61402496.

REFERENCES

[1] L. Barroso, J. Clidaras, and U. Hoelzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," vol. 8, no. 3, p. 154, 2009.
[2] A. Rabkin and R. Katz, "How hadoop clusters break," *IEEE Software*, vol. 30, no. 4, pp. 88–94, 2013.
[3] MySQL, http://www.mysql.com/.
[4] Httpd, http://httpd.apache.org/.
[5] Y. Y. Su, M. Attariyan, and J. Flinn, "Autobash: improving configuration management with operating system causality analysis." in *ACM Symposium on Operating Systems Principles 2007, SOSP 2007, Stevenson, Washington, Usa, October*, 2007, pp. 362–371.
[6] J. Mickens, M. Szummer, and D. Narayanan, "Snitch: interactive decision trees for troubleshooting misconfigurations," in *Usenix Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*, 2007, p. 8.
[7] M. Attariyan and J. Flinn, "Automating configuration troubleshooting with dynamic information flow analysis," in *Usenix Conference on Operating Systems Design and Implementation*, 2010, pp. 1–11.
[8] Z. Dong, M. Ghanavati, and A. Andrzejak, "Automated diagnosis of software misconfigurations based on static analysis," in *IEEE International Symposium on Software Reliability Engineering Workshops*, 2013, pp. 162–168.
[9] T. Xu, X. Jin, P. Huang, Y. Zhou, S. Lu, L. Jin, and S. Pasupathy, "Early detection of configuration errors to reduce failure damage," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16. Berkeley, CA, USA: USENIX Association, 2016, pp. 619–634. [Online]. Available: http://dl.acm.org/citation.cfm?id=3026877.3026925
[10] J. Zhang, L. Renganarayana, X. Zhang, N. Ge, V. Bala, T. Xu, and Y. Zhou, *EnCore: exploiting system environment and correlation information for misconfiguration detection*. ACM, 2014.
[11] A. Rabkin and R. Katz, "Static extraction of program configuration options," in *International Conference on Software Engineering*, 2011, pp. 131–140.
[12] S. Zhou, X. Liu, S. Li, W. Dong, X. Liao, and Y. Xiong, "Confmapper: automated variable finding for configuration items in source code," in *International Conference on Software Quality, Reliability and Security-Companion*, 2016.
[13] Clang, https://clang.llvm.org/.
[14] LLVM, https://www.llvm.org/.
[15] T. Xu, J. Zhang, P. Huang, J. Zheng, T. Sheng, D. Yuan, Y. Zhou, and S. Pasupathy, "Do not blame users for misconfigurations," in *Twenty-Fourth ACM Symposium on Operating Systems Principles*, 2013, pp. 244–259.
[16] Y. Xiong, A. Hubaux, S. She, and K. Czarnecki, "Generating range fixes for software configuration," in *International Conference on Software Engineering*, 2012, pp. 58–68.
[17] R. Pandita, X. Xiao, H. Zhong, T. Xie, S. Oney, and A. Paradkar, "Inferring method specifications from natural language api descriptions," in *International Conference on Software Engineering*, 2012, pp. 815–825.
[18] V. Raychev, M. Vechev, and E. Yahav, "Code completion with statistical language models," *Acm Sigplan Notices*, vol. 49, no. 6, pp. 419–428, 2014.
[19] M. Pradel and T. R. Gross, "Name-based analysis of equally typed method arguments," *IEEE Transactions on Software Engineering*, vol. 39, no. 39, pp. 1127–1143, 2013.

Combing Data Filter and Data Sampling for Cross-Company Defect Prediction: An Empricial Study

Xiao Yu[1,2,3], Man Wu[2,3], Yan Zhang[2,3*], Mandi Fu[4]
[1]State Key Lab. of Software Engineering, Computer School, Wuhan University, Wuhan, China
[2]School of Computer Science and Information Engineering, HuBei University, Wuhan, China
[3]Educational Informationalization Engineering Research Center of HuBei Province, Wuhan, China
[4]Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, China
*Corresponding author email: zhangyan@hubu.edu.cn

***Abstract*—Cross-company defect prediction (CCDP) is a practical way that trains a prediction model by exploiting one or multiple projects of a source company and then applies the model to target company. Unfortunately, larger irrelevant cross-company (CC) data usually makes it difficult to build a prediction model with high performance. On the other hand, the CC data has the highly imbalanced nature between the defective-prone and non-defective classes, which will degrade the performance of CCDP. To address such issues, this paper proposes an approach, in which data sampling is combined with data filter, to overcome these problems. Data sampling seeks a more balanced dataset through the addition or removal of instances, while data filter is a process of filtering out the irrelevant CC data so that the performance of CCDP models can be improved. We employ two data filtering methods called NN filter and DBSCAN filter combined with SMOTE (Synthetic Minority Oversampling Technique) and RUS (Random Under-Sampling). Eight different approaches would be produced when combing these four techniques: 1- NN filter performed prior to RUS; 2- NN filter performed after RUS; 3- NN filter performed prior to SMOTE; 4- NN filter performed after SMOTE; 5- DBSCAN filter performed prior to RUS; 6- DBSCAN filter performed after RUS; 7- DBSCAN filter performed prior to SMOTE; 8- DBSCAN filter performed after SMOTE. The empirical study was carried out on 15 publicly available project datasets. The experimental results demonstrate that NN filter performed prior to RUS (Approach 1) performs better than the other seven approaches.**

Keywords—software defect prediction;cross-company defect prediction;data sampling; data filter

I. INTRODUCTION

Software defect prediction is one of the most important software quality assurance techniques. It aims to detect the defect proneness of new software modules via learning from defect data. So far, many efficient software defect prediction approaches [1-6] have been proposed, but they are usually confined to within project defect prediction (WPDP). WPDP works well if sufficient data is available to train a defect prediction model. However, it is difficult for a new project to perform WPDP if there is limited historical data. Cross-company defect prediction (CCDP) is a practical approach to solve the problem. It trains a prediction model by exploiting one or multiple projects of a source company and then applies the model to target company [7]. In recent years, most existing CCDP approaches have been proposed. The challenges of building a CCDP model with high performance usually include:

(1) How to weaken the impact of irrelevant CC data to improve the performance of CCDP.

The ability to transfer knowledge from a source company to a target company depends on how they are related. The stronger the relationship, the more usable will be CC data. The performance of CCDP is generally poor because of larger irrelevant CC data. Previous work [8] founds that using raw CC data directly would increase false alarm rates due to irrelevant instance in CC data, so several data filtering works should be done before building the prediction model. For example, Turhan et al. [8] and Peters et al. [9] proposed the NN filter and the Peters filter to select the CC instances which are mostly similar to WC data as the training dataset.

(2) How to cope with the class imbalance problem to improve the performance of CCDP.

The CC data has the highly imbalanced nature, because the number of non-defective instances is usually larger than the number of non-defective ones. The class imbalance problem may cause difficulties for learning, as most classification algorithms only perform optimally when the number of instances of each class is roughly the same. When these algorithms are trained by a highly skewed dataset in which the minority class is heavily outnumbered by the majority class, these classifiers tend to favor the majority class and have less ability to classify the minority class. Therefore, several data sampling works should be done before building the CCDP model. For example, Lin et al. [10] introduced a novel CCDP approach named Double Transfer Boosting (DTB). DTB firstly uses NN filter to filter out irrelevant CC data, and then uses SMOTE algorithm [11] to re-sample the CC data before building the CCDP model.

However, it is still unclear what extent combining data filter and data sampling can contribute to CCDP, and how to make better use of them to improve CCDP. To address such issues, this paper proposes an approach, in which data sampling is combined with data filter, to cope with both class imbalance problem and the influence of irrelevant CC data. A question may arise when we combine the two techniques, that is, which technique, data filter or data sampling, should be performed first? To answer the question, we employ two data filtering methods called NN filter [8] and DBSCAN filter [12] combined with SMOTE [11] and RUS [13]. We investigate

DOI reference number: 10.18293/SEKE2017-134

eight different approaches: 1- NN filter performed prior to RUS; 2- NN filter performed after RUS; 3- NN filter performed prior to SMOTE; 4- NN filter performed after SMOTE; 5- DBSCAN filter performed prior to RUS; 6- DBSCAN filter performed after RUS; 7- DBSCAN filter performed prior to SMOTE; 8- DBSCAN filter performed after SMOTE. To our best knowledge, no study has been done for combining data filter with data sampling and investigating the eight approaches in the domain of CCDP.

The empirical study was carried out over 15 publicly available projects, all of which exhibit a high degree of class imbalance between the defective-prone and non-defective classes. Three different learners were used to build CCDP models. The experimental results demonstrate that NN filter performed prior to RUS (Approach 1) has significantly better performance than the other seven approaches.

The remainder of this paper is organized as follows. Section II presents the related work. Section III describes two data filtering methods, two data sampling methods, and eight combination approaches. Section IV demonstrates the experimental results. Section V discusses the potential threats to validity. Finally, Section VI addresses the conclusion and points out the future work.

II. Related Work

In this section, we first review the existing cross-company and cross-project defect prediction approaches. Then, we briefly review the data sampling methods.

A. Defect prediction

In order to solve the problem that the new companies have too limited historical data to perform WCDP well, the cross-project and cross-company defect prediction appear. Zimmermann et al. [55] studied CCDP models on 12 real-world applications datasets. Their results indicated that CCDP is still a serious challenge because of the different distribution between the training project data and the target project data. In order to narrow the distribution gap, there are three mainstream ways.

The first one is to apply the data filtering method to find the best suitable training data (e.g., [8, 9, 12, 14]). For example, Turhan et al. [8] proposed a nearest neighbor (NN) filter to select cross-company data. Peters et al. [9] introduced the Peters filter to select training data.

The second mainstream way is to design effective defect predictor based on transfer learning techniques (e.g., [7, 10, 15,16, 17, 18, 19]). For instance, Ma et al. [15] proposed Transfer Naïve Bayes (TNB) model. Chen et al. [10] proposed double transfer boosting (DTB) model. Another challenge in CCDP is that the set of metrics between the source company data and target company data is usually heterogeneous. Jing et al. [7] and Chen et al. [19] proposed the effective solutions for heterogeneous cross-company defect prediction.

The third mainstream way is to apply unsupervised classifier that does not require any training data to perform CCDP (e.g., [20-23]), therefore the distribution gap between the training project data and the target project data is no longer an issue. For instance, Zhang et.al [22] proposed to apply a connectivity-based unsupervised classifier that is based on spectral clustering to perform CPDP.

B. Data sampling

Besides the larger irrelevant instances in CC data, CCDP models also suffer from the class imbalance problem. A considerable amount of research has been done to investigate this problem at data and algorithm levels. Data-level methods include a variety of resampling techniques, manipulating training data to rectify the skewed class distributions, such as random oversampling and random under-sampling. They are simple and efficient, but their effectiveness depends greatly on the problem and training algorithms [24]. Algorithm-level methods address class imbalance by modifying their training mechanism directly with the goal of better accuracy on the minority class, including one-class learning [25], and cost-sensitive learning algorithms [26]. Algorithm-level methods require specific treatments for different kinds of learning algorithms, which hinders their use in many applications, because we do not know in advance which algorithm would be the best choice in most cases. In addition to the aforementioned data-level and algorithm-level solutions, ensemble learning [27] has become another major category of approaches to hand imbalanced data by combining multiple classifiers, such as SMOTEBoost [28], and AdaBoost.NC [29]. Ensemble learning algorithms have been shown to be able to combine strength from individual learners, and enhance the overall performance. They also offer additional opportunities to handle class imbalance at both the individual and ensemble levels. This paper investigates SMOTE and RUS, because of their simplicity, effectiveness, and popularity in the literature.

While a great deal of work has been done for data filter and data sampling separately, limited research has been done and reported on both together, especially in the context of CCDP. Among the few studies, Lin et al. [17] proposed the combination of NN filter and SMOTE to preprocess the CC data before building the CCDP model. However, it is still unclear what extent combining data filter and data sampling can contribute to CCDP, and how to make better use of them to improve CCDP. This is exactly what we will solve in this paper.

III. Methodology

A. Data filter

Previous work [8] founds that using raw CC data directly would increase false alarm rates due to irrelevant instances in CC data, so several data filtering works should be done before building the prediction model. The main goal of data filter is to select the most valuable training data for the CCDP model by filtering out irrelevant instances in CC data. In this study, we employ two representative data filtering methods, NN filter and DBSCAN filter.

The NN filter was proposed by Turhan et al. [8]. Based on the widely used KNN algorithm, NN filter can find out the most similar $k \times n$ instances from CC data while n is the number of WC instances and k is the parameter of the KNN algorithm. The procedure of NN filter is as follows:

1. Find k neighbors from the CC data for each WC instance based on Euclidean distance, and

2. Collect the selected neighbors without duplication into a new CC data.

The DBSCAN filter was proposed by Kawata et al. [12]. It assumes that CC instances which are in the same cluster as WC instances are the most valuable instances in CC data. The procedure of DBSCAN filter is as follows:

1. Combine the CC data and WC data,

2. Find sub-clusters by using DBSCAN algorithm,

3. Select sub-clusters which consist at least one WC instance, and

4. Collect the CC data in the selected sub-clusters into a new CC data.

B. Data sampling

Besides larger irrelevant instances in CC data, CCDP models also suffer from the class imbalance problem. A variety of data sampling techniques have been studied in the literature. The main goal of data sampling is to achieve a certain balance between the defective-prone class and non-defective class. In this study, we employ two representative data sampling methods, SMOTE (Synthetic Minority Oversampling Technique) and RUS (Random Under-Sampling).

SMOTE [11] was proposed by Chawla in 2002. It is an over-sampling approach in which the minority class is over-sampled by creating "synthetic" examples. The procedure of SMOTE is as follows:

1. For each instance in the defective-prone class, calculate the Euclidean distance between it and other instances in the defective-prone class to find its *k* nearest neighbor.

2. According to the amount of over-sampling, determine the sampling rate and select a certain number of instances from *k* nearest neighbor randomly. The sampling rate (between defective-prone and non-defective instances) was set to 50:50 throughout the experiments.

3. Take the difference of the feature vector between it and its nearest neighbor.

4. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration.

5. Generate new instances for each instance in the defective-prone class and add new samples into it.

RUS (Random Under-Sampling) [13] is a simple method to select a subset of non-defective instances randomly and then combine them with defective-prone instances as a training set.

The procedure of random under-sampling is as follows:

1. Calculate the ratio of the defective-prone class to the non-defective class, and get the sampling frequency.

2. Sample the non-defective class by the sampling frequency.

3. Select all defective-prone instances.

4. Combine selected instances and attributes for training.

C. Eight combination approaches

The primary goal of this study is to evaluate the data preprocessing technique in which data filter is combined with data sampling. Eight different scenarios (also called approaches) would be produced depending on whether data filter is performed before or after data sampling. The eight different approaches are described as follows:

- Approach 1: NN filter performed prior to RUS;
- Approach 2: NN filter performed after RUS;
- Approach 3: NN filter performed prior to SMOTE;
- Approach 4: NN filter performed after SMOTE;
- Approach 5: DBSCAN filter performed prior to RUS;
- Approach 6: DBSCAN filter performed after RUS;
- Approach 7: DBSCAN filter performed prior to SMOTE;
- Approach 8: DBSCAN filter performed after SMOTE.

IV. EXPERIMENTS

In this section, we evaluate the eight combination approaches to perform CCDP empirically. We first introduce the experiment dataset, the performance measures and the experimental procedure. Then, in order to investigate the performance of the eight combination approaches, we perform some empirical experiments to find answers to the research question mentioned above.

A. Data set

In this experiment, we employ 15 available and commonly used datasets which can be obtained from PROMISE. The 15 datasets have the same 20 attributes, so we can apply all attribute information directly. Table 1 tabulates the details about the datasets.

TABLE I. DETAILS OF EXPERIMENT DATASET

Project	***Examples***	***%Defective***	***Description***
ant	125	16	Open-source
arc	234	11.5	Academic
camel	339	3.8	Open-source
elearn	64	7.8	Academic
jedit	272	33.1	Open-source
log4j	135	25.2	Open-source
lucene	195	46.7	Open-source
poi	237	59.5	Open-source
prop	660	10	Proprietary
redaktor	176	15.3	Academic
synapse	157	10.2	Open-source
systemdata	65	13.8	Open-source
tomcat	858	9	Open-source
xalan	723	15.2	Open-source
xerces	162	47.5	Open-source

B. Performance measures

In the experiment, we employ three commonly used performance measures including *pd*, *pf* and *g-measure*. They are defined in Table 2 and summarized as follows.

TABLE II. PERFORMANCE MEASURES

		Actual	
		yes	***no***
Predicted	***yes***	***TP***	***FP***
	no	***FN***	***TN***
pd	$\frac{TP}{TP+FN}$		
pf	$\frac{FP}{FP+TN}$		
G-measure	$\frac{2*pd*(1-pf)}{pd+(1-pf)}$		

- Probability of detection or *pd* is the measure of defective modules that are correctly predicted within the defective class. The higher the *pd*, the fewer the false negative results.

- Probability of false alarm or *pf* is the measure of non-defective modules that are incorrectly predicted within the non-defective class. Unlike *pd*, the lower the *pf* value, the better the results.

- *G-measure* is a trade-off measure that balances the performance between *pd* and *pf*. A good prediction model should have high *pd* and low *pf*, and thus leading to a high *g-measure*.

C. Experimental Procedure

In every experiment, one dataset is selected as WC data and the rest are regarded as CC data to conduct the experiment. The CC data is considered as basic training data which will be adjusted in every experiment. All processing steps (data filter and data sampling) are done on CC data. Then processed CC data are used to build the CCDP model. Finally, the resulting model is evaluated on the WC data. The procedure will be repeated 30 times in every experiment to avoid sample bias. Then, the mean values of performance are calculated.

In this experiment, we choose three representative classifiers as the basic prediction model, Naive Bayes (NB) [33], Random Forest (RF) [34], and Logistic Regression (LR) [35].The reason we choose these classifiers is that these classifiers fall into three different families of learning methods. NB is a probabilistic classifier; RF is a decision-tree classifier; and LR is a linear model for classification.

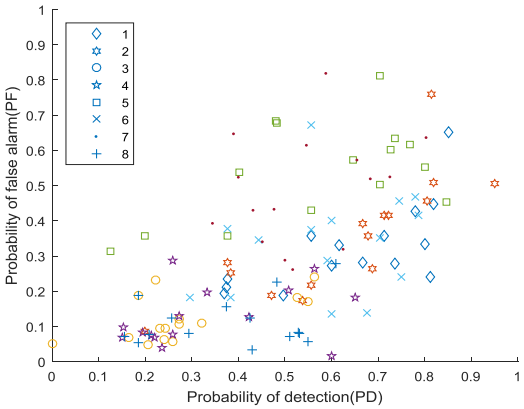
D. Experimental results

Fig. 1 presents the scatter plots of (PD, PF) points from the eight approaches on the fifteen projects. Note that a CCDP approach has more points distributed at bottom right if it has higher PD value and lower PF value. We can gain the following results from Fig. 1.

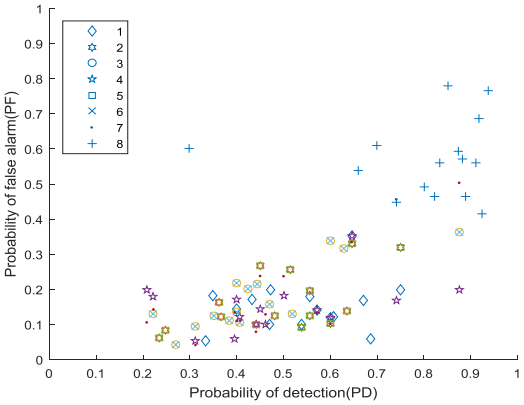
(1) In terms of the defect detection rate (PD), Approach 1 outperforms other approaches for NB model, which shows its effectiveness in finding defects. However, Approach 1 has a little high PF value. In terms of the false alarm rate (PF), although Approach 3 is the best, it performs not well in PD, which makes it hardly useful in practice.

(2)Although Approach 8 appears to be better at PD than other approaches for RF model, it performs the worst in PF. Approach 1 presents generally higher PD and lower PF than other approaches for RF model.

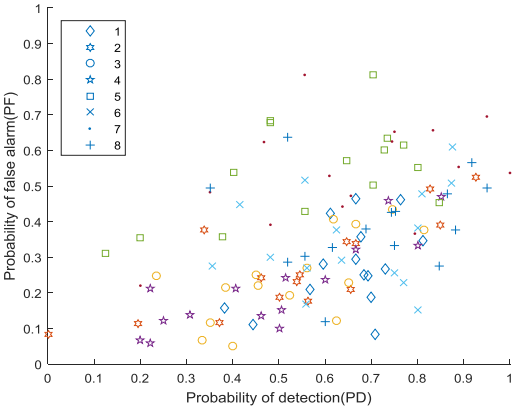
(3) Approach 8 shows better PD than other approaches for LR model, but its advantage is rather limited. Approach 1 presents generally higher PD and lower PF than other approaches for RF model.



(a)NB



(b)RF



(c)LR

Fig. 1. Performances with Scatter plots of (PD, PF) points of the eight approaches on the fifteen projects

Tables 3-5 show the G-measure values for each project on all approaches with three CCDP models.

TABLE III. G-MEASURE PERFORMANCES WITH NAÏVE BAYES

Project	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
ant	**0.64**	0.56	0.53	0.56	0.50	0.57	0.60	0.54
arc	0.66	0.62	0.56	0.56	0.63	**0.65**	0.52	0.63
camel	0.67	0.68	0.54	0.60	0.72	**0.72**	0.54	0.71
elearn	0.71	0.72	0.55	0.72	0.62	**0.75**	0.31	0.72
jedit	0.57	0.52	0.46	0.55	0.59	**0.65**	0.55	0.59
log4j	0.62	0.59	0.60	0.60	0.65	**0.80**	0.58	0.63
lucene	0.64	0.51	0.50	0.54	0.54	**0.67**	0.57	**0.67**
poi	0.49	0.37	0.42	0.47	0.55	**0.62**	0.45	0.57
prop	0.49	**0.61**	0.55	0.60	0.58	**0.66**	0.50	0.59
redaktor	0.61	**0.66**	**0.66**	0.63	0.35	0.45	0.29	0.34
synapse	**0.77**	0.71	0.74	0.63	0.37	0.54	0.65	0.43
system	0.66	0.68	0.57	0.35	0.67	0.65	0.51	**0.71**
tomcat	0.69	**0.73**	0.65	0.69	0.58	0.62	0.45	0.60
xalan	**0.67**	0.66	0.63	0.66	0.47	0.52	0.52	0.51
xerces	**0.51**	0.39	0.35	0.34	0.34	0.48	0.37	0.36
AVG	**0.63**	0.60	0.55	0.57	0.54	**0.62**	0.50	0.57

TABLE IV. G-MEASURE PERFORMANCES WITH RANDOM FOREST

Project	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
ant	**0.73**	0.65	0.66	0.72	0.57	0.60	0.60	0.59
arc	0.51	**0.65**	0.41	0.40	0.58	0.67	0.51	0.31
camel	0.64	0.51	0.37	0.26	**0.66**	0.52	0.54	0.37
elearn	0.66	0.33	0.00	**0.75**	0.31	0.71	0.31	0.50
jedit	**0.69**	0.66	0.47	0.57	0.48	0.53	0.55	0.27
log4j	**0.73**	0.60	0.34	0.38	0.49	0.76	0.58	0.45
lucene	0.51	**0.65**	0.28	0.36	0.51	0.44	0.57	0.34
poi	0.62	**0.65**	0.38	0.35	0.39	0.67	0.45	0.57
prop	**0.67**	0.64	0.38	0.26	0.52	0.65	0.50	0.40
redaktor	**0.50**	0.37	0.30	0.38	0.30	0.41	0.29	0.30
synapse	**0.78**	0.71	0.65	0.64	0.21	0.75	0.55	0.52
system	0.60	**0.64**	0.34	0.47	0.56	0.59	0.51	0.50
tomcat	**0.66**	0.65	0.42	0.42	0.43	0.63	0.55	**0.66**
xalan	**0.66**	0.62	0.64	0.62	0.51	0.63	0.52	0.59
xerces	**0.50**	0.49	0.42	0.32	0.38	0.47	0.37	0.53
AVG	**0.63**	0.59	0.40	0.46	0.46	0.60	0.49	0.46

TABLE V. G-MEASURE PERFORMANCES WITH LOGISTIC REGRESSION

Project	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
ant	**0.74**	0.71	0.71	0.73	0.57	0.70	0.46	0.66
arc	**0.61**	0.52	0.49	0.36	0.58	0.57	0.54	0.60
camel	0.72	0.57	0.52	0.45	0.66	0.77	0.63	**0.78**
elearn	0.73	0.50	0.56	0.33	0.31	**0.82**	0.32	0.71
jedit	0.66	**0.72**	0.63	0.67	0.48	0.48	0.58	0.65
log4j	**0.77**	0.62	0.61	0.64	0.49	0.67	0.70	**0.73**
lucene	0.59	0.63	0.56	0.54	0.51	0.63	0.59	**0.64**
poi	0.53	0.65	0.63	0.60	0.39	0.63	0.50	**0.65**
prop	**0.66**	0.63	0.57	0.61	0.52	**0.66**	0.49	0.65
redaktor	**0.69**	0.63	0.71	0.65	0.30	0.52	0.28	0.43
synapse	0.72	0.67	0.73	0.39	0.21	**0.75**	0.47	0.71
system	0.59	0.66	0.64	0.67	0.56	0.54	0.59	**0.62**
tomcat	0.59	0.32	0.50	0.63	0.43	0.62	0.42	**0.65**
xalan	0.63	0.63	**0.64**	0.62	0.51	0.63	0.53	0.59
xerces	**0.60**	0.44	0.36	0.34	0.38	0.47	0.42	0.41
AVG	**0.66**	0.56	0.59	0.55	0.46	0.63	0.50	0.63

We can gain the following results from Tables 3-5.

(1) On more than half projects, Approach 6 performs better than the other seven approaches for NB model. However, Approach 1 achieves the best average G-measure value for NB model (see Table 4).

(2) On nine projects, Approach 1 achieves higher G-measure than the other seven approaches for RF model. In addition, Approach 1 achieves the best average G-measure value for RF model (see Table 5).

(3) On six projects, Approach 1 achieves higher G-measure than the other seven approaches for LR model, while Approach 8 displayed similar or slightly worse performance than Approach 1. In addition, Approach 1 achieves the best average G-measure value for LR model (see Table 5).

Therefore, we can conclude that NN filter performed prior to RUS (Approach 1) has significantly better performance than the other seven approaches.

V. THREATS TO VALIDITY

In this section, we discuss several validity threats that may have an impact on the results of our studies.

External validity. Threats to external validity occur when the results of our experiments cannot be generalized. As a preliminary result, we performed our experiments on the 15 datasets to answer the research questions. Although these datasets have been widely used in many software defect prediction studies, we still cannot claim that our conclusions can be generalized to other software projects. Nevertheless, this work provides a detail experimental description, including parameter settings (default parameter settings specified by *sklearn*), thus other researchers can easily replicate this empirical study on new datasets.

Internal validity. Threats to internal validity refer to the bias of the choice of CCDP classifiers, data filtering methods and data sampling methods. In this work, we only use three classifiers, Naive Bayes (NB), Random Forest (RF), and Logistic Regression (LR) due to its popularity in defect prediction. In addition, we choose two representative data filtering methods, i.e., NN filter and DBSCAN filter, two representative data sampling methods, i.e., RUS and SOMTE.

Construct validity. Threats to construct validity focus on the bias of the measures used to evaluate the performance of CCDP. In our experiments, we mainly use pd, pf, G-measure to measure the effectiveness of the eight approaches. Nonetheless, other evaluation measures such as AUC measure can also be considered.

VI. CONCLUSION AND FUTURE WORK

In this paper, we propose data filter combined with data sampling to overcome the influence of larger irrelevant CC data and class imbalance problems that often affect CCDP models. Eight combination approaches are investigated: 1- NN filter performed prior to RUS; 2- NN filter performed after RUS; 3- NN filter performed prior to SMOTE; 4- NN filter performed after SMOTE; 5- DBSCAN filter performed prior to RUS; 6- DBSCAN filter performed after RUS; 7- DBSCAN filter performed prior to SMOTE; 8- DBSCAN filter performed after SMOTE.

We conduct experiments on the 15 datasets to evaluate the performance of the proposed eight approach. The experimental results indicate that NN filter performed prior to RUS (Approach 1) has significantly better performance than the other seven approaches.

In the future, we would like to employ more datasets to validate the generalization of the derived conclusions [36-37].

ACKNOWLEDGMENT

This work is partly supported by Educational Informationalization Engineering Research Center of HuBei Province.

REFERENCES

[1] Elish K O, Elish M O. Predicting defect-prone software modules using support vector machines[J]. Journal of Systems and Software, 2008, 81(5): 649-660.

[2] Zheng J. Cost-sensitive boosting neural networks for software defect prediction[J]. Expert Systems with Applications, 2010, 37(6): 4537-4543.

[3] Sun Z, Song Q, Zhu X. Using coding-based ensemble learning to improve software defect prediction[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012, 42(6): 1806-1817.

[4] Wang S, Yao X. Using class imbalance learning for software defect prediction[J]. IEEE Transactions on Reliability, 2013, 62(2): 434-443.

[5] Liu M, Miao L, Zhang D. Two-stage cost-sensitive learning for software defect prediction[J]. IEEE Transactions on Reliability, 2014, 63(2): 676-686.

[6] Jing X Y, Ying S, Zhang Z W, et al. Dictionary learning based software defect prediction[C]//Proceedings of the 36th International Conference on Software Engineering. ACM, 2014: 414-423.

[7] Jing X, Wu F, Dong X, et al. Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning[C]//Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. ACM, 2015: 496-507.

[8] Turhan B, Menzies T, Bener A B, et al. On the relative value of cross-company and within-company data for defect prediction[J]. Empirical Software Engineering, 2009, 14(5): 540-578.

[9] Peters F, Menzies T, Marcus A. Better cross company defect prediction[C]//Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on. IEEE, 2013: 409-418.

[10] Chen L, Fang B, Shang Z, et al. Negative samples reduction in cross-company software defects prediction[J]. Information and Software Technology, 2015, 62: 67-77.

[11] Chawla N V, Bowyer K W, Hall L O, et al. SMOTE: synthetic minority over-sampling technique[J]. Journal of artificial intelligence research, 2002, 16: 321-357.

[12] Kawata K, Amasaki S, Yokogawa T. Improving relevancy filter methods for cross-project defect prediction[M]//Applied Computing & Information Technology. Springer International Publishing, 2016: 1-12.

[13] Tahir M A, Kittler J, Yan F. Inverse random under sampling for class imbalance problem and its application to multi-label classification[J]. Pattern Recognition, 2012, 45(10): 3738-3750.

[14] Herbold S. Training data selection for cross-project defect prediction[C]//Proceedings of the 9th International Conference on Predictive Models in Software Engineering. ACM, 2013: 6.

[15] Ma Y, Luo G, Zeng X, et al. Transfer learning for cross-company software defect prediction[J]. Information and Software Technology, 2012, 54(3): 248-256.

[16] Nam J, Pan S J, Kim S. Transfer defect learning[C]//Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013: 382-391.

[17] Chen L, Fang B, Shang Z, et al. Negative samples reduction in cross-company software defects prediction[J]. Information and Software Technology, 2015, 62: 67-77.

[18] Ryu D, Choi O, Baik J. Value-cognitive boosting with a support vector machine for cross-project defect prediction[J]. Empirical Software Engineering, 2016, 21(1): 43-71.

[19] Cheng M, Wu G, Jiang M, et al. Heterogeneous Defect Prediction via Exploiting Correlation Subspace[J].

[20] Bishnu P S, Bhattacherjee V. Software fault prediction using quad tree-based k-means clustering algorithm[J]. IEEE Transactions on knowledge and data engineering, 2012, 24(6): 1146-1150.

[21] Catal C, Sevim U, Diri B. Metrics-driven software quality prediction without prior fault data[M]//Electronic Engineering and Computing Technology. Springer Netherlands, 2010: 189-199.

[22] Zhang F, Zheng Q, Zou Y, et al. Cross-project defect prediction using a connectivity-based unsupervised classifier[C]//Proceedings of the 38th International Conference on Software Engineering. ACM, 2016: 309-320.

[23] Zhong S, Khoshgoftaar T M, Seliya N. Unsupervised Learning for Expert-Based Software Quality Estimation[C]//HASE. 2004: 149-155.

[24] Estabrooks A, Jo T, Japkowicz N. A multiple resampling method for learning from imbalanced data sets[J]. Computational intelligence, 2004, 20(1): 18-36.

[25] Japkowicz N, Myers C, Gluck M. A novelty detection approach to classification[C]//IJCAI. 1995, 1: 518-523.

[26] Zhou Z H, Liu X Y. Training cost-sensitive neural networks with methods addressing the class imbalance problem[J]. IEEE Transactions on Knowledge and Data Engineering, 2006, 18(1): 63-77.

[27] Rokach L. Ensemble-based classifiers[J]. Artificial Intelligence Review, 2010, 33(1-2): 1-39.

[28] Chawla N V, Lazarevic A, Hall L O, et al. SMOTEBoost: Improving prediction of the minority class in boosting[C]//European Conference on Principles of Data Mining and Knowledge Discovery. Springer Berlin Heidelberg, 2003: 107-119.

[29] Wang S, Yao X. Negative Correlation Learning for Class Imbalance Problems School of Computer Science, University of Birmingham[J]. 2012.

[30] Lewis D D. Naive (Bayes) at forty: The independence assumption in information retrieval[C]//European conference on machine learning. Springer Berlin Heidelberg, 1998: 4-15.

[31] Hall T, Beecham S, Bowes D, et al. A systematic literature review on fault prediction performance in software engineering[J]. IEEE Transactions on Software Engineering, 2012, 38(6): 1276-1304.

[32] Boetticher G, Menzies T, Ostrand T. PROMISE Repository of empirical software engineering data[J]. West Virginia University, Department of Computer Science, 2007.<http://promisedata.org/repository>.

[33] Lewis D D. Naive (Bayes) at forty: The independence assumption in information retrieval[C]//European conference on machine learning. Springer Berlin Heidelberg, 1998: 4-15.

[34] Breiman L. Random forests[J]. Machine learning, 2001, 45(1): 5-32.

[35] Hosmer Jr D W, Lemeshow S, Sturdivant R X. Introduction to the Logistic Regression Model[J]. Applied Logistic Regression, 2000.

[36] Liu Z, Wei C, Ma Y, et al. UCOR: an unequally clustering-based hierarchical opportunistic routing protocol for WSNs, International Conference on Wireless Algorithms, Systems, and Applications. Springer Berlin Heidelberg, 2013: 175-185.

[37] Liu Z, Wei C, Ma Y, et al. UCOR: an unequally clustering-based hierarchical opportunistic routing protocol for WSNs, International Conference on Wireless Algorithms, Systems, and Applications. Springer Berlin Heidelberg, 2013: 175-185.

SnippetGen: Enhancing the Code Search via Intent Predicting

Qing Huang[1], Xudong Wang[2], Yangrui Yang[3], Hongyan Wan[1], Rui Wang[1], Guoqing Wu[1*],
[1] State Key Laboratory of Software Engineering, Computer School,Wuhan University, Wuhan, China
[2] International School of Software, Wuhan University, Wuhan, China
[3] College of Information Engineering of North China University of Water Resources and Electric Power, Zhengzhou, China
Email:[1]{qh, why0511, wangrui1989, wgq}@whu.edu.cn, [2]hsu@whu.edu.cn, [3]yangyangrui@ncwu.edu.cn

***Abstract*—To enable the cod sarch results to run immediately without any subsequent modification, an intent-enhanced code search approach (IECS) is proposed. It has the ability of intent predicting to guess what else a user might do after obtaining the search results. Based on the intent-relevant semantic and structural matches, IECS improves the performance of code search by incorporating the intent for expansion. To perform IECS, the code search tool SnippetGen is implemented. Compared with CodeHow and Google Code Search (CS), SnippetGen outperforms them by 28.5% with a precision score of 0.846 (i.e., 84.6% of the first results are relevant).**

Keywords-Code search; Intent predicting; Query expansion

I. Introduction

To reuse the existing method, many code search tools are proposed. Early code search engines, e.g., Google[1], Krugle[2] and Koders[3], offered only the keyword-based search with low precision. The later work did semantic search to enhance the accuracy of the search results, e.g., signature matching, type matching [15], [10]. But these approaches were impractical, because they required too little or too much specification. The current work supports query expansion to promise the better usability, such as CodeHow [2]. It considers the impact of the APIs and expands the query with the APIs. Although these existing code search tools seem to yield correct matches, the search results could not meet the user demands directly and need to be modified [3]. One major reason is that these tools lack the ability of intent predicting to guess what else a user might do after obtaining the search results.

Example: Given a query "access data in excel", "the method ExcelToDataSet" could be returned by the standard Boolean model [1], because the method contains all the query terms with the highest term frequency such as " Excel8.0; " and "Fill (*dataset*) ". Unfortunately this method is always modified, because it only accesses the data from the outdated excel2003 but fails in the most frequently-used excel2007. In this case, a user changes this method from (*"Microsoft.Ace.OLEDB.4.0", "Excel8.0"*) to (*"Microsoft.Ace.OLEDB.12.0", "Excel12.0"*). If search engine could anticipate this intent, it can generate an expansion query "Microsoft.Ace.OLEDB.12.0 Excel12.0 access data excel" and retrieve the modified method. This example shows that through intent predicting , more accurate code search could be achieved.

In this paper, we propose an intent-enhanced code search approach (*IECS*) using the intent to enhance the search. Figure 1 presents the overall structure containing intent-enriched, intent-first and intent-expanded component . In the intent-enriched component (*as shown in Fig.1a*), code modifications are recorded through the code version tracking service . Then an intent extraction algorithm is proposed to exploit the intents from modifications. This algorithm refines the commonly-used intents, extracts the intent-relevant context and enriches methods with the intents and context in turn. In the intent-first component, results are retrieved by computing the semantic and structural similarity scores between the intent and the query , and combining the two similarity scores (*as shown in Fig.1b*). If the results cannot match the query, the intent-expanded component is triggered to expand a query by considering both the intent and the text similarity (*as shown in Fig.1c*). Finally, the Extended Boolean model [17] is adopted to retrieve the more relevant results.

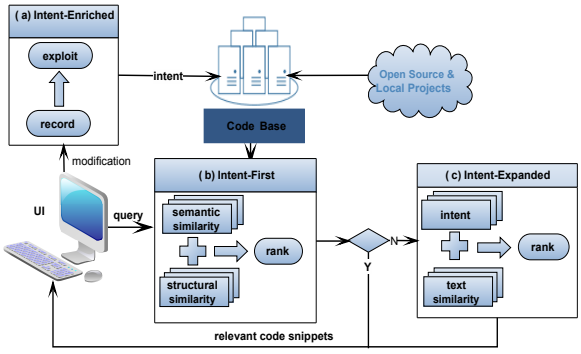


Fig. 1. Similar changes to three methods

An IECS-supported code search tool SnippetGen is implemented. The front-end is a visual studio 2010 extension. The backend is a codebase which is constructed by collecting 2,151

[1]codesearch.google.com
[2]krugle.org
[3]koders.com

*Corresponding author
DOI reference number:10.18293/SEKE2017-037

projects from Github[4] and indexing 1.16 million C# methods using Lucene[5]. SnippetGen is compared with CodeHow [2] and Google Code Search (CS) [16] by performing 70 real-world queries. The results show that SnippetGen achieves a precision score of 0.846 which outperforms them by 28.5% when the top 1 results are inspected. The results con?rm the effectiveness of SnippetGen in programming practices.

The contributions of this paper are as follows:

- An intent-enhanced code search approach (IECS) is proposed. It contains an intent extraction algorithm to exploit the potential intents of the method.
- IECS performs code retrieval preferentially within the intent scope based on semantic and structural matches, which is more effective than within the complete method scope.
- SnippetGen, that performs IECS, is implemented. The experiment results show that SnippetGen outperforms CodeHow and CS by 28.5% with a precision score of 0.846.

II. Intent-enriched component

An intent extraction algorithm is proposed to extract the intents from the past modifications. Algorithm 1 describes this process procedurally in the four steps. Note that each method may have much intent. If SnippetGen incorporates all intent for expansion, it may produce worse results than not expanding the query. Thus the refinement strategy is considered to ensure the intents that benefit the code search.

Step 1: Identifying modifications. For each method m_i, SnippetGen employs ChangeDistiller [11] to compare the AST of the m_i's old and new versions from the past modifications. Then SnippetGen characterizes modifications as a sequence of node operations Δ_i consisting of node insertions, deletions, updates and moves.

Step 2: Refining modifications. SnippetGen uses the modified Longest Common Edit Operation Subsequence (LCEOS) algorithm [18] to identify the common node operations pairs $\Delta_c = \bigcap_{i=1}^{n} \Delta_i$, such that $\forall 1 \ll i \ll n, \Delta_{c_i} \subseteq \Delta_i$, by iteratively comparing the node operations pairwise. In these common node operations pairs, if one or more concrete instances of types, methods, variables and constants have the same edit type or inheriting type, despite of the different name, SnippetGen thinks they are abstract equivalent. Then it generalizes these concrete instances with abstract identifiers \$$t$, \$$m$, \$$v$ and \$$c$, so as to enforce a consistent naming. Meanwhile, it records the mapper between the abstract identifiers and the concrete instances. If some subsequent node operations pairs are inconsistent with the current mapper, they are omitted.

Step 3: Extracting Intents. SnippetGen extracts the intents from the mapper. Meantime, it extracts the intent-relevant context with control, data and containment dependence analysis. This context comprises the unchanged AST nodes that depend on the node operations or on which node operations depend in common node operations Δ_c.

[4]https://github.com/explore

[5]http://lucene.apache.org/

Algorithm 1 IntentExtraction

Input: Original Version O, a set of Modified Version M
Output: Intents
/* step 1: identify modificaion */
foreach m_i in M
  Δ_i=m_i-o //obtain the i-th modifications(Δ_i)
  add Δ_i to Δ_s; // obtain all modifications(Δ_s)
end
/* step 2: refine modification */
// identify the common modifications
Δ_c= $\bigcap_{i=1}^{n} \Delta_i$ such that $\forall 1 \leq$ i $\leq$ n $\Delta_{ci} \subseteq \Delta_i$
foreach Δ_{c_i} in Δ_c
  // obtain the node operations pairs(nop) from Δ_{c_i}
  $\Delta_{c_i} \rightarrow$ nop:
  // obtain the concrete instances(ci) of types, methods, variables and constants from nop
  nop$\rightarrow$ ci
  if abstractMatch(ci) is true
    // ci's edit type or inheriting type is equivalent
    if ci is inconsistent with the mapper
      omit nop;
      continue;
    end
    ci=ai;//substitute the abstract identifiers(ai) for ci
    build the mapper(ai,ci);
  end
end
/* step 3 extract intents */
mapper$\rightarrow$ intents; //extract the intents from the mapper
method = method+intents; //enrich the method with the intents
Return intents

Step 4: Having obtaining intents, SnippetGen enriches each method with the intents and intent-relevant context.

Example: Given the method ExcelToDataSet's original version O, there are two modified versions F, S. SnippetGen employs ChangeDistiller to input (O, F) and output Δ_F= $\{Update\,(O_1\ F_1)\}$, which shows the variable "*Provider*" and the variable "*Extended Properties*" (line1) are changed to "*OleDb.12.0*" and "*Excel12.0; HDR=NO*" respectively (*as shown in Fig.2a*). Similarly, SnippetGen inputs (O, S) and outputs Δ_B=$\{Update\,(O_1\ S_1)\}$ (*as shown in Fig.2b*). Thus SnippetGen identifies the longest common node operations pairs, such that Δ_c=$\Delta_F \cap \Delta_S$=$\{\text{pair}_1\,(\text{Update}\,(O_1\ F_1)\,, \text{Update}\,(O_1\ S_1))\}$. In these common node operations pairs, SnippetGen substitutes the abstract identifiers \$$v_1$ and \$$v_2$ for ("*Microsoft.ACE.OLE DB.12.0*", "*Excel12.0; HDR=NO*") in Δ_A,("*Microsoft.ACE. OLEDB.12.0*", "*Excel12.0;*") in Δ_B (*as shown in Fig.2c*). Meanwhile, it records the mapper ($\$v_1, \$v_2$)=$\{$("*Microsoft. Ace.OleDb.12.0*", "*Excel12.0; HDR=NO*"), ("*Microsoft.Ace. OleDb.12.0*", "*Excel12.0*"). Finally, SnippetGen extracts the intent from this mapper , such that Intent=("*Microsoft.ACE. OLEDB.12.0, Excel12.0; HDR=NO*").

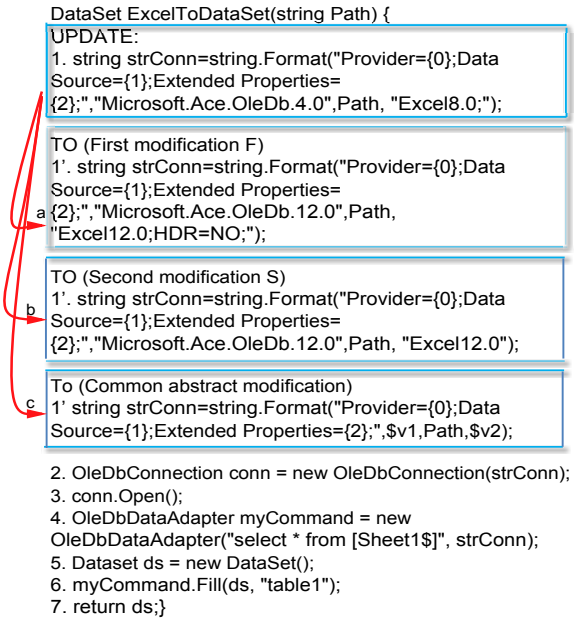


Fig. 2. Similar changes to three methods

III. Intent-first component

To retrieve the relevant methods preferentially within the method's intent scope, SnippetGen computes the semantic similarity between the query and the intents as well as the structural similarity between the query and the intent-relevant contexts, combines the two similarity values, and finally returns the relevant code. In this process, we define two method scores and an operation.

Definition 1 (Semantic Score). For each method m_i, SnippetGen views the query and m_i's intents, as a bag of words, computes their textual similarity score m_i^d. score using VSM[6] [9], and returns the top i semantic method scores denoted as m^d:

$$m^d = \{m_1^d,\ m_2^d,\ \cdots,\ m_i^d\}$$

In VSM, the query and the m_i's intent are represented by a vector. The term frequency (*tf*) and inverse document frequency (*idf*) are calculated based on the frequency of words.

Definition 2 (Structural Score). For each method m_i, SnippetGen views the query and m_i's intent-relevant context as a bag of function calls, computes their structural similarity score m_i^s. score using VSM, and returns the top i structural method scores denoted as m^s:

$$m^s = \{m_1^s,\ m_2^s,\ \cdots, m_i^s\}$$

In VSM, the query and the method are represented by a vector. *Tf* and *idf* are calculated based on the frequency of the function being called.

Definition 3 (Score Combination). Let the methods appearing in both m^d and m^s as $m_{overlap}$, and let the methods appearing only in m^d or m^s as $m_{notoverlap}$. Given m_i^d and m_i^s, the combination as follows:

$$m_i.\text{score} = \begin{cases} m_i^d.\text{score} + m_i^s.\text{score} & (\text{if } m_i \in m_{overlap}) \\ \dfrac{\text{MinOverlapScore} \times m_i^{d/s}.\text{score}}{\text{maxNotOverlapScore} + \propto} & (\text{if } m_i \notin m_{overlap}) \end{cases} \quad (1)$$

where *MinOverlapScore* is the minimum score of all methods in $m_{overlap}$; *maxNotOverlapScore* is the maximum score of all methods in $m_{notoverlap}$. If m_i only appears in m^d, $m_i^{d/s}$.score equals to m_i^d.score. If m_i only appears in m^s, $m_i^{d/s}$.score equals to m_i^s.score. The parameter $\propto$ is an adjustment factor to make sure that the score of $m_{overlap}$ is larger than that of $m_{notoverlap}$. Empirically, we set $\propto$ to 0.1.

Actually, Equation (1) says that, if m_i appears in both m^d and m^s, its score is the sum of the two scores. Otherwise, its score is calculated based on the similarity score in m^d or m^s. SnippetGen computes the scores in above way and obtains the top i potentially relevant methods:

$$m_{relevant} = \{m_1, m_2, \cdots, m_i\}$$

Example: for the query "access data in excel", the semantic relevant methods m^d with similarity scores are {*Excel2007 ToDateSet=0.5, AccessToDateSet=0.4, ExcelTo DateSet=0.4*}. The structural methods m^s with similarity scores are {*OleDbDataAdapter.Fill=0.9, Excel2007ToDateSet=0.6, ExcelToDateSet=0.5*}.

The overlapping methods $m_{overlap}$ are "*Excel2007To DateSet*" and "*ExcelToDateSet*". We compute their score as $0.5 + 0.6 = 1.1$, and $0.4 + 0.5 = 0.9$, respectively.

The non-overlapping methods $m_{notoverlap}$ are "*OleDbDataAdapter.Fill*" and "*AccessToDateSet*". We get Min OverlapScore value 0.9 and maxNotOverlapScore value 0.9. Thus the scores for "*OleDbDataAdapter.Fill*" and "*AccessToDateSet*" are 0.81 and 0.36, respectively. Finally, the rank of potentially relevant methods $m_{relevant}$ are as follows:

- *"Excel2007ToDateSet" (score=1.1);*
- *"ExcelToDateSet" (score=0.9);*
- *"OleDbDataAdapter.Fill" (score=0.81);*
- *"AccessToDateSet" (score=0.36).*

IV. Intent-expanded component

If the methods retrieved by intent-first component cannot match the query, following the query expansion option [7], SnippetGen expand a query with intents to retrieve the relevant methods.

A query Q_t containing n terms is defined as:

$$Q_t = \ (t_1, \cdots t_n)$$

For a method, three features is defined as:

$$F = (f_1, f_2, f_3)$$

where f_1 stands for the intent ; f_2 stands for the FQN; f_3 stands for the method body .

[6]https://github.com/hcutler/tf-idf/tree/c505c72af7f3eb6e3dd5b10d9e8f54c08e1434d3

A query can be expressed in terms of f_i: t_i where $t_i \in Q_t$ and $f_i \in F_t$. It means to search for methods that contains the term t_i in a field f_i. SnippetGen constructs a Boolean query expression for retrieving methods that match the query in terms of text similarity:

$$q_{text} = (f_2 : t_1 \vee f_3 : t_1) \wedge \cdots \wedge \ \ (f_2 : t_n \vee f_3 : t_n)$$

This query expression searches for methods that contain the terms $t_1, \ldots, t_n$ in fields f_2 (*FQN*) and f_3 (*Method Body*).

After the intent-first component, SnippetGen gets k potentially relevant methods $m_{relevant}$. For each method m_i in $m_{relevant}$, SnippetGen tokenizes the intent to get a keyword list A_i. Then it constructs Boolean query expressions as follows:

$$q_{m_i} = f_1 : \ intent_i \wedge \ \ (f_2 : t_1 \vee f_3 : t_1) \\ \wedge \cdots \wedge \ \ (f_2 : t_n \vee f_3 : t_n)$$

where $m_i \in m_{relevant}$ and $t_k \in (Q_t - A_i)$. Note that we remove the terms that appear in A_i from the query Q_t, since the impact of these terms have been considered in the m_i's intent . This query searches for the methods that contain the intent$_i$ in fields f_1(*intent*) as well as other query terms in fields f_2(*FQN*) and f_3(*Method Body*).

A method may be retrieved by more than one query expressions defined above . SnippetGen combines the query expressions into an expanded query for retrieving methods:

$$q_{expand} = (q_{m_1}, q_{m_2}, \cdots, q_{m_k}, q_{text})$$

Given the query: "access data in excel", the query terms Q_t=(*access, data, excel*). The potentially relevant method's intent is ("*Microsoft,Ace,OleDb,12.0,Excel12.0,HDR=NO*"). The comprehensive query expressions are as follows:

q_{m_1}=$(f_1$:$Microsoft, Ace, OleDb, 12.0, Excel12.0$
$HDR = NO) \wedge (f_2$:$access \vee f_3 : access)$
$\wedge (f_2 : data \vee f_3 : data)$
q_{text}=$(f_2 : access \vee f_3 : access) \wedge (f_2 : date \vee f_3 : date)$
$\wedge (f_2 : excel \vee f_3 : excel)$

To retrieve relevant methods given the queries, we adopt the Extended Boolean model (EBM) [17], which combines the characteristics of the VSM and Boolean model. Given a query expression $q_{expand} = (q_{m_1}, \cdots, q_{m_k}, q_{text})$, it is easy to implement EMB by using Lucene[7].

V. Experiment

A. Setup

First, a codebase as the backend of SnippetGen is constructed by collecting 2,151 projects downloaded from Github and indexing 1.16 million C# methods by using Lucene. Second, 70 real-world queries are employed by Portfolio's author [8]. All these queries are formulated as set of keywords to address some programming tasks reported in Portfolio's user study. Third, the front-end of SnippetGen as a Microsoft Visual Studio 2010 extension is implemented.

[7]https://lucene.apache.org/core/2_9_4/scoring.html

To investigate the effectiveness of our approach, 20 participants are employed. Six participants are graduate students who have at least of two years of C# programming experience. The others are PhD students who have 3-6 years of C# programming experience. Each participant runs SnippetGen, CodeHow and Google Code Search (CS) to address 3-4 queries and inspect the top 20 results for each query to judge whether they are relevant or not.

CodeHow is the latest query-expanded code search tool [2]. To reprogram it, we index the online MSDN[8] document as expansion library using Lucene. Participants use CodeHow and enter the query directly to retrieve the results. CS represents conventional keywords-based code search web applications. Participants should go to the website, look for implementations and extract them by copying and pasting results into the workspace.

B. Evaluation Metrics

To evaluate the effectiveness of SnippetGen, we make use of the Precision@k[9]:

$$\text{Precision@}k = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{\text{relevant}_{i,k}}{k} \qquad (2)$$

where $relevant_{i,k}$ represents the relevant methods for query i in the top k results, Q is a set of queries. Precision@k takes an average on all queries whose relevant answers could be found by inspecting the top $k(k = 1, 5, 10, 20)$ results. A better code search tool allows developers to discover the needed code by examining fewer results. The higher the metric value, the higher the accuracy is.

We also make use of Normalized Discounted Cumulative Gain (NDCG)[10] [12] to measure the ranking capability of the code search based on the graded relevance of the results of a set of queries. It varies from 0.0 to 1.0, with 1.0 representing the ideal ranking of the results. The higher the NDCG value , the better the ranking capability is.

C. Experimental Results

We compare SnippetGen with CodeHow and CS by performing the 70 queries. As Table I shows, when the top 1 results are inspected, SnippetGen achieves a precision score of 0.846, which means that 84.6% of the first results are relevant methods without any subsequent modification. When the top 5 results are inspected, SnippetGen achieves a precision score of 0.861. These results are considered satisfactory . Note that only the results, which both receive relevant feedback and need not to be modified subsequently, are labeled as relevant. Thus the precision of CodeHow and CS is lower than previous papers, such as ref [2], [10].

We pick out CodeHow for comparative analysis, as it is the latest code search tool proposed in 2015. As Table I shows,

[8]https://msdn.microsoft.com/en-us/library

[9]https://github.com/jcnewell/MyMediaLiteJava/blob/master/src/org/mymedialite/eval/measures/PrecisionAndRecall.java

[10]https://github.com/jcnewell/MyMediaLiteJava/blob/master/src/org/mymedialite/eval/measures/NDCG.java

CodeHow achieves a score of 0.658 when the top 1 results are inspected. SnippetGen achieves 28.5%, 66.2%, 70.3%, and 89.5% improvements in terms of Precision@1, Precision@5, Precision@10, and Precision@20 , respectively. In terms of NDCG, SnippetGen obtains a score of 0.873, which also outperforms the CodeHow (0.712) by 22.6%. In the same way, SnippetGen performs better than CS.

TABLE I
THE COMPARISON AMONG SNIPPETGEN, CODEHOW AND CS

	SnippetGen	CodeHow	CS
Precision@1	0.846	0.658	0.421
Precision@5	0.861	0.518	0.368
Precision@10	0.792	0.465	0.346
Precision@20	0.762	0.402	0.283
NDCG	0.873	0.712	0.682

Figure 3 shows the percentage of queries that SnippetGen performs better/worse than CodeHow. When the top 1 results are examined, SnippetGen wins in 36% of the queries and loses in 18% of the queries. In terms of the top 5 results, SnippetGen wins in 61% of the queries and loses in only 7% of the queries. The results confirm that the improvement achieved by SnippetGen is significant.

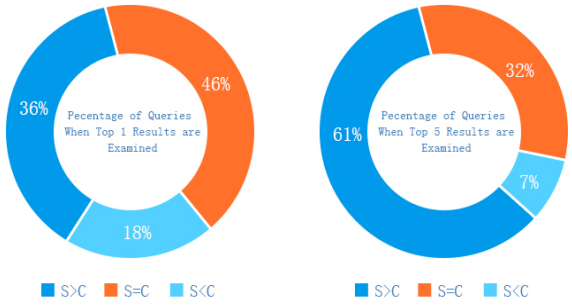


Fig. 3. The mechanism of change pattern

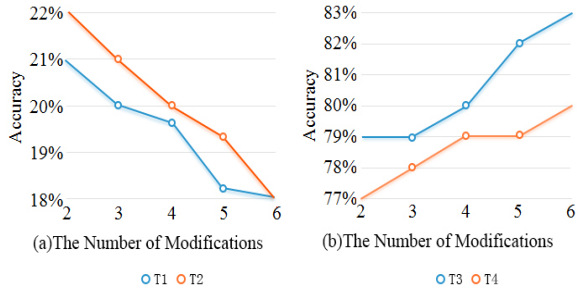


Fig. 4. The mechanism of change pattern

To analyze the reason for the lost cases, we continue to explore what factors are correlated with the accuracy of SnippetGen. We depict the two lost cases (i.e., T1 "Convert utc time to local time" and T2 "How to get Color from Hexadecimal color code") as shown in Fig.4a. They show that the more past modifications provided, the less common subset is likely to be shared among these modifications. This results in that the accuracy goes down. For the draw, the methods are never modified. In this case, SnippetGen should be similar to what other tools could achieve. Different from Fig.4a, we depict the two promising cases (i.e., T3 "access data from excel" and T4 "find regular LINQ expressions") as shown in Fig.4b. They show that the more past modifications provided, the higher accuracy is. It illustrates when methods are similar, adding modifications may not decrease the number of common modifications, but may induce more identifier abstraction and produce more sufficient intents.

These results illustrate that the accuracy of SnippetGen varies with the similarity and number of past modifications. Even the similarity takes precedence over the number. Too many modifications or too few is not good for the accuracy. The more similar modifications are, the higher accuracy is.

VI. DISCUSSIONS

Intent Source: The intent is extracted from each method's past modifications. No modifications, no intent to occu r. It results in SnppetGen not matching semantic similarity between the query and the method's intent, as well as not matching structural similarity between the query and the method's intent-relevant context. According to our user study, the methods without intent make up 25.7% to 38.4% of methods' volume in codebase. If the method carries no intent, SnippetGen uses the original FQN and Method Body.

Intent Sensitivity: The intent is closely related to past modifications, but the intent varies inconsistently with the number of the modifications. The more modifications are provided, the fewer common subset is likely to be shared, which results in the problems of over generalization. The fewer modifications are provided, the more common subset is, which results in the problems of over specification. Actually, instead of the number of the modifications, the intent strictly depends on the similarity among them. If the modifications are diverse, SnippetGen extracts the fewer common modifications and obtains the insufficient intent. If the modifications are similar, SnippetGen extracts the more common modifications and obtains the sufficient intent.

To reach the maximum similarity, we try two ways. First, we apply the heuristic algorithm to pick out the similar modifications from all modifications, so that the intent becomes more sufficient despite a big difference between a few modifications. Second, we use the threshold in LCEOS (as shown in the process of identifying common intent) to tolerate inexact matches among modifications. For example, if SnippetGen fails to find any common edit operation between two modifications, it generalizes all concrete instances of types, methods, variables and constants with abstract identifiers to match edit type or inheriting type.

VII. RELATED WORK

Early code search engines are the keywords-based information retrieval techniques [4]. For example , Google, Koders and Krugle allow a user to input keywords and perform the file-level retrieval. However, Myers [16] observed that these

code search engines always achieve inaccurate search results, because they are not designed to support programming tasks. To improve the accuracy, later work did semantic search. Originally, the work by Wing looked at matching function signatures [15]. Then it was extended to match more complete formal semantics using λ prolog and Larch-based [19]. But these techniques were impractical because either they attempted to do too little or too much. Recently, to promise better usability, several approaches have been proposed to improve the effectiveness of free-text code search via query expansion. For a vague input, adding one or more synonyms of the words appearing in the query can enhance the precision of search results. McMillan proposed Portfolio that takes natural language descriptions as "synonyms" and outputs a list of functions or code fragments along with corresponding call graphs [6]. Wang et al. [5] proposed an active code search approach which incorporates user feedback as "synonyms" to refine the query. Fei et al. [2] propose a latest code search technique that could understand the APIs a user query refers to and considers both text similarity and potential APIs. In addition, there are other query expansions either by using an appropriate ontology [20], natural language [14], or collaborative feedback [13].

These existing code search tools seem to yield semantically correct matches, but the search results might be too complex or too slow to meet the user needs. These results still has to be modified. But SnippetGen can retrieve the more relevant methods without any subsequent modification. Although our work is viewed as a query expansion, we differ from them. We give the code search engine the ability of intent predicting to guess what else the user might do after he obtains the search results. We incorporate intent as "synonyms" for expansion and consider the impact of both potential intents and text similarity on code search. Besides, we propose the refinement strategy in the intent extraction algorithm. This strategy can pick out the appropriate intents to benefit the code search.

VIII. Conclusion

In this paper, an intent-enhanced code search approach (IECS) is proposed. Based on the intent-relevant semantic and structural matches, it exploits the intent before performing code retrieval and allows a user to retrieve the relevant code by expanding the query with the intent. In the future, we plan to address the issues discussed in Section VII. For example, in the intent-enriched component, we either improve similarityCchoosing heuristic algorithm to ensure a sufficient intent, or employ the deep learning approach to make the intent become self-improvement.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Projects No. 61373039, No. 61170022, No. 61003071 and No. 91118003.

References

[1] H. Niu, I. Keivanloo, and Y. Zou, "Learning to rank code examples for code search engines," Empirical Software Engineering, pp. 1-33, 2016.

[2] F. Lv et al., "CodeHow: Effective Code Search Based on API Understanding and Extended Boolean Model (E)." IEEE/ACM International Conference on Automated Software Engineering pp. 260-270.2015

[3] J. Galenson et al., "CodeHint: dynamic and interactive synthesis of methods." International Conference on Software Engineering pp. 653-663.2014

[4] P. Fafalios, and Y. Tzitzikas, "Post-analysis of Keyword-Based Search Results Using Entity Mining, Linked Data, and Link Analysis at Query Time." IEEE International Conference on Semantic Computing pp. 36-43.2014

[5] S. Wang, D. Lo, and L. Jiang, "Active code search: incorporating user feedback to improve code search relevance." Acm/ieee International Conference on Automated Software Engineering pp. 677-682.2014

[6] C. Mcmillan et al., "Portfolio: Searching for relevant functions and their usages in millions of lines of code," Acm Transactions on Software Engineering & Methodology, vol. 22, no. 4, pp. 402-418, 2013.

[7] C. Carpineto, and G. Romano, "A Survey of Automatic Query Expansion in Information Retrieval," Acm Computing Surveys, vol. 44, no. 1, pp. 159-170, 2012.

[8] C. McMillan, M. Grechanik, D. Poshyvanyk, Q. Xie, and C. Fu. Portfolio: finding relevant functions and their usage. In ICSE, 2011.

[9] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval: The Concepts and Technology behind Search. Addison-Wesley, 2011.

[10] T. J. Biggerstaff, B. G. Mitbander, and D. E. Webster, "Program understanding and the concept assignment problem." Communications of the Acm pp. 482-498.2010

[11] B. Fluri, M. Wursch, M. Pinzger, and H. C. Gall. "Change distillingł tree differencing for fine-grained source code change extraction. IEEE Transactions on Software Engineering", 33 (11):18, November 2007.

[12] E. Linstead et al., "Sourcerer: mining and searching internet-scale software repositories," Data Mining & Knowledge Discovery, vol. 18, no. 2, pp. 300-336, 2009.

[13] Taciana A. Vanderlei, Frederico A. Durao, Alexandre C. Martins, Vinicius C. Garcia, Eduardo S. Almeida, and Silvio R. de L. Meira, "A cooperative classification mechanism for search and retrieval software components," Proc SAC'07, pp. 866-871 (March 2007).

[14] Christopher G. Drummond, Dan Ionescu, and Robert C. Holte, "A learning agent that assists the browsing of software libraries," IEEE Trans. on Software Engineering Vol. 26(12) pp. 1179-1196 (December 2000)

[15] A. M. Zaremski, and J. M. Wing, "Signature matching: a key to reuse," Acm Sigsoft Software Engineering Notes, vol. 18, no. 5, pp. 182-190, 1993.

[16] Y. S. Maarek, D. M. Berry, and G. E. Kaiser, "An information retrieval approach for automatically constructing software libraries," IEEE Transactions on Software Engineering, vol. 17, no. 8, pp. 800-813, 1991.

[17] G. Salton, E. A. Fox, and H. Wu, "Extended boolean information retrieval," Commun. ACM, vol. 26, pp. 1022C1036, 1983.

[18] J. W. Hunt and T. G. Szymanski. "A fast algorithm for computing longest common subsequences". CACM, 20 (5):350C353, 1977.

[19] 31. Eugene J. Rollins and Jeannette M. Wing, "Specifications as search keys for software libraries," Proc. 8th Intl. Conf. on Logic Programming, pp. 173-187 (1991).

[20] Haining Yao and Letha Etzkorn, "Towards a semanticbased approach for software reusable component classification and retrieval," ACMSE'04, pp. 110-115 (April 2004).

LaSaS: an Aggregated Search based Graph Matching Approach

Ghizlane ECHBARTHI
Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, France
Email: ghizlane.echbarthi@univ-lyon1.fr

Hamamache KHEDDOUCI
Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, France
Email: hamamache.kheddouci@univ-lyon1.fr

***Abstract*—Graph querying is crucial to fully exploit the knowledge within the widely used graph datasets. However, graph datasets are usually noisy which makes the *approximate graph matching* tools favored to overcome restrictive query answering. In this paper, we introduce a new framework of approximate graph matching based on aggregated search called Label and Structure Similarity Aggregated Search (LaSaS). LaSaS enables effective and efficient graph querying without considering any fixed schema of the data graph by *(i)* using the aggregated search strategy to increase the number of answers, *(ii)* using a lightweight graph similarity metric that takes into account nodes label and graph structure similarity to enable finding approximate matches and also by *(iii)* using a simple graph weight update routine instead of computing the maximum common subgraph which reduces the overall computation cost. We evaluated our proposed approach over the real-life DBpedia graph and results show the effectiveness and stability of the approach on different parameter settings. Moreover, results also show that LaSaS yields more precise matches in a shorter amount of time when compared to state-of-the-art related approaches.**

***Keywords*—Graph querying, Graph matching, Graph similarity metric, Aggregated search.**

I. Introduction

Nowadays, a steep increase in data production is witnessed, which favors the use of graphs as a storage support to fully exploit the knowledge within the dataset. In fact, graphs are a popular data model that enables efficient data processing benefiting from all the graph theory findings and technics. In considering this matter, graph querying has attracted the interest of many researchers as it represents an essential task to exploring the knowledge in these datasets.

In order to query these graph databases, generally, a *graph matching* task is performed using either *graph isomorphism* to find exact answers to the query, or other technics that allows *approximate graph matching*. In the case of graph isomorphism, seeking exact answers to the query can be very expensive as well as restrictive since actual datasets are usually noisy. Moreover, it requires the user to have a complete knowledge of the data structure which is not always the case. To bypass these restrictions, approximate graph matching is widely used in many real life applications such as web anomaly detection [1], search result classification [2] and spam detection [3] to name a few.

However, few works on graph querying have addressed the graph matching problem by building an answer to the query based on the aggregation of heterogeneous graphs. The idea is to find a matching to a given query by combining several subgraphs that when aggregated, they form an approximate match for the query. This search paradigm is known as aggregated search in graphs [4], [5] *i.e.*, given a query graph q and a set B of graphs, aggregated search aims to find matches to q by combining or aggregating subgraphs in B. The aggregated search is different from the classical graph matching problem where all occurrences of the query are to be found in one target graph G. In contrast to the graph context, *aggregated search* is a popular search paradigm in Information Retrieval where an answer to a query is given by combining contents of heterogeneous sources, *e.g.*, text, image, and video instead of giving the classical list of relevant links. The concept of aggregation has rarely been tackled in graph databases, though it represents various advantages compared to the classical graph matching problem. Furthermore, aggregated graph matching represents a powerful matching paradigm, as it brings about significant improvements when considering the number of findable solutions to a given query, where a simple graph matching can not find any. A motivating example for the aggregated search in graph databases is plagiarism detection: the aggregated graph search can detect a plagiarism even when the cheater takes several small shards from different documents and combines them. The aggregated graph search can detect these plagiarism cases.

Generally, most works on graph querying adhere to costly operations such as graph isomorphism [6], [7], or maximum common subgraph search [4], which are too restrictive. Similarly, approximate matching frameworks consist in either strict label similarity [8], or structural similarity [9] which restricts the number of findable solutions. Moreover, to the best of our knowledge, approximate graph matching using the aggregated search paradigm is novel and has never been tackled.

Broadly motivated by addressing these shortcomings, we propose in this paper a new framework for approximate graph matching called Label and Structure Similarity Aggregated Search (LaSaS). The proposed framework enables an effective RDF graph querying without any knowledge of neither the de facto SPARQL language nor the schema of the data graph. Our contribution is threefold: First, we use the aggregated search paradigm to enrich the set of answers. Second, a lightweight graph similarity metric that takes into account both the graph label and graph structure similarity to enable finding approximate matches. Third, we propose a simple graph weight update that replaces the maximum common subgraph search task and reduces the complexity cost.

The remaining of the paper is organized as follows: Section

DOI reference number: 10.18293/SEKE2017-046

2 presents an overview of the related work about graph querying. In Section 3, we present preliminaries and necessary notions for the understanding of our LaSaS method, which is entirely presented in Section 4. Section 5 reports and discusses our experimental results. Section 6 concludes with a summary of our contributions and raises issues for future work.

II. Related Work

Graph querying is a crucial task as most of the actual datasets are being stored and exploited as graphs. Many graph querying techniques exist in the literature and the proposed framework is closely related to aggregated search and approximate (sub)graph matching, the latter being considered as an elementary operation in our matching framework. So in this section, we provide a brief description of the graph querying methods that are based on (sub)graph matching and the aggregated search in graphs.

Subgraph matching. Subgraph matching is a well-studied problem with a rich literature. Two main categories fall under the subgraph matching problem, the *exact* and the *inexact subgraph matching*. Exact subgraph matching finds exact answers to the query via graph isomorphism [7], [6]. Approaches in this group are criticized for their intractability [10], their cost prohibitive characteristic and for being restrictive *w.r.t.* to the number of answers to the query.

On the other hand, inexact graph matching allows slight differences between the query and the matches which is highly suitable for actual needs as graph datasets are usually noisy, and relevant answers could be found using approximate matching. Under the category of inexact (sub)graph matching, we find subgraph matching based on graph simulation [11], [12], which can be determined in quadratic time and is defined as a *relation* between the query nodes and the target nodes. However, it suffers a considerable loss of structural similarity which makes it untailored for many applications such as in bioinformatics.

Another variant of inexact graph matching is the graph homomorphism [13]. The idea is to find mappings to the query such that node labels difference falls under a threshold whereas the query edges are mapped to paths of a given length.

More related to our work, Tian *et al.* proposed a tool called SAGA [14]. The approach allows approximate matching by breaking the query into subgraphs and finding small hits that are assembled to form a match. Although SAGA method is based on an efficient indexing to speed up the processing, it actually reduces to solving the maximum clique problem which represents a costly operation.

It is also worth mentioning that there are parallel works related to the approximate graph matching in bioinformatics. This category of methods includes PathBlast [15], NetAlign [16] and IsoRank [17] to name but a few. These methods are found to be efficient in tasks related to the domain application such protein interaction networks alignment [16], [15].

Besides, a different type of approximate structural matching works has been proposed in [18], [8]. These works are based on concept propagation [19] and spreading activation [20] instead of classical matching schemes (graph isomorphism and similarity metric). Nevertheless, these works consider a strict label node matching, which is still too restrictive.

Although not pursued here, approximate and inexact (sub)graph matching encompasses decades of research work. Hence, we encourage interested readers to [21] as well as references therein for further details about the problem.

Querying RDF graphs. In the realm of querying RDF graphs, SPARQL is a widely used query language. It requires a complete knowledge of the schema of the graph database, *i.e.*, the structure, node labels and types of entities in the graph. Moreover, writing queries in SPARQL proves to be a complicated task that requires users to be familiar with the language. In order to alleviate this, Zou *et al.* studied the problem of answering SPARQL queries via subgraph matching and proposed gStore [9], which allows approximate node label matching but adheres to strict structural matching leading the method to be restrictive.

Aggregated search in graph databases. Aggregated search is a familiar search paradigm in Information Retrieval in the context of documents [22]. However, few works tackled the problem of aggregated search in graph databases. Elghazel *et al.* [4], [5] studied the problem of aggregated search in labeled graphs and their problem formulation is close to ours. Nevertheless, their method looks for exact matches which is restrictive on one hand, and on the other hand very expensive due to the maximum common subgraph search and the maximum clique detection, which are considered as being among the most expensive operations on graphs.

Most of the aforementioned works altogether lack several critical points: (*i*) They present a restrictive tool for query answering either by imposing a strict node label similarity or by a strict structural similarity, which does not allow approximate relevant answers to be discovered, (*ii*) the graph matching is usually done by checking the maximum common subgraph which is a costly computational task. (*iii*) Methods querying semi-structured data requires a complete knowledge of the graph schema and are often complicated. (*iv*) Few works considered a query answering by several graphs in concomitant, and if so, they seek exact matches which is still restrictive. To address these shortcomings, we propose a new graph matching framework that aims to answer a query by allowing approximate label and structural similarity through a lightweight similarity metric which alleviate the task of maximum common subgraph search. Moreover, our proposed framework enables efficient RDF graph querying without having any knowledge about the schema of the graph. Last but not least, our proposed framework is based on the aggregated search to enable a joint query answering by several heterogeneous graphs in order to benefit from the information wealth within these graphs.

III. Preliminaries

This section is dedicated to introducing notions used in our proposed method. First, we give definitions about the usual data structures used: the query, the target graphs and the answer set. Then we introduce a new graph similarity metric and the objective function of the proposed matching scheme, we end this section by giving the problem formulation of aggregated grah search.

A. Query, Target graph, Answer

Query. The query q is an undirected labeled graph $q = (V_q, E_q, \mu_q)$ where V_q represents the vertex set, and E_q the edge set. μ_q is a function $\mu_q : V_q \rightarrow L_{V_q}$ that associates labels to vertices, with L_{V_q} the vertex label set.

Target graphs. Target graphs are represented by the set B of p graphs which are referred to as fragments f_i, $i \in [p]$. Fragments $f_i = (V_i, E_i, \mu_i)$ are undirected labeled graphs *s.t.* V_i (resp. E_i) is the vertex set (resp. edge set) of f_i. μ_i is a function $V_i \rightarrow L_{V_i}$ associating labels to vertices, with L_{V_i} is the vertex label set of f_i.

Answer set. The expected answer set A is defined as follows: $A = \{a_1, \cdots, a_k\}$ where a_i are called aggregates. We have $a_i = \bigcup_{j=0}^{p_i} f_{i_j}$ *s.t.* p_i is the number of fragments in a_i and we have $p_i \leq p$, *i.e.*, a_i is constituted by as few fragments as possible.

Though our method focuses on undirected labeled graphs, it is straightforward to extend it to process other kinds of graphs.

B. Objective function

Graph similarity metric. We first introduce a graph similarity metric that computes the similarity between two graphs based on two main features: *(i)* the node label similarity and *(ii)* the structure similarity. The similarity function is denoted by $s(f_i, q)$ where $s : B \times \{q\} \rightarrow [0, 1]$ is a function that quantifies the similarity that f_i shares with q in terms of label and structure similarity *s.t.* the closest s to 1, the more similar are f_i and q. In the following, we present the components of the similarity metric s: Label similarity and structure similarity components.

1) Label similarity component: Label similarity of fragment f_i and query q is denoted by $\Delta_L(f_i, q)$ and is computed as follows:

$$\Delta_L(f_i, q) = \frac{1}{n_i}. \sum_{v \in V_{f_i}} J(\mu(v), \mu(Q(v))) \quad (1)$$

where n_i is the number of vertices in f_i, and J is the jaccard similarity coefficient such that $J(l_1, l_2) = \frac{W_{l_1} \cap W_{l_2}}{W_{l_1} \cup W_{l_2}}$, with W_{l_1} (resp. W_{l_2}) is the words set in label l_1 (resp. l_2). Q is an application $Q : V_i \rightarrow V_q$ that associates vertices from the fragment to their counterpart in the query and we have: $Q(v) = u$ iff $J(\mu_i(v), \mu_q(u)) > \tau$ where τ is a user fixed threshold.

2) Structure similarity component: On the other hand, $\Delta_D(f_i, q)$ denotes the structure similarity between fragment f_i and query q as follows:

$$\Delta_D(f_i, q) = \frac{1}{n_i}. \sum_{u,v \in V_{f_i}, u<v} \delta_t(d(u, v), d(Q(u), Q(v))) \quad (2)$$

where

$$\delta_t(x, y) \begin{cases} 1 & \text{if } |x - y| < t \\ 0 & \text{otherwise.} \end{cases}$$

$d(u, v)$ denotes the distance between vertices u and v. Δ_D computes the structure similarity in terms of distances in f_i and q. Δ_D increases each time the distance between two nodes in the target graph is equal or near to the distance between their corresponding nodes in the query graph. More precisely, Δ_D increases when the distance difference is under a threshold t called the *structure disparity threshold*. The value assigned to threshold t should be small in order to avoid disparate matches and to preserve the structure similarity. Besides, t should not neither be equal to zero otherwise it would be very restrictive as only strikingly similar matches will be favored.

Taking these both components into account, we define the similarity function s as follows:

$$s(f_i, q) = \lambda.\Delta_L(f_i, q) + (1 - \lambda).\Delta_D(f_i, q) \quad (3)$$

where λ is a tunable parameter in $[0, 1]$.

Objective function. Let η denote the matching that associates the aggregates a_i in the answer set A to the query q (matching query nodes to target nodes). We define the objective function of the matching η as follows:

$$\Psi(\eta) = \frac{1}{k}. \sum_{i=0}^{k} s(a_i, q) \quad (4)$$

C. Problem Formulation

Given a set of fragments B, a query q. The aggregated graph search problem consists in finding the matching η that associates the answer set A of k aggregates to q *s.t.* $\Psi(\eta) = 1$.

Proposition 1. *Aggregated graph search problem is NP-Hard.*

Proof: Let us consider the simple case of $k = 1$. That is to say, the expected answer set consists of one aggregate a_1 *s.t.* $a_1 = \bigcup_{j=0}^{p_1} f_j$, where $\Psi(\eta) = 1$, *i.e.*, $s(a_1, q) = 1$, with η is the matching that associates aggregate a_1 to q. By definition, the similarity function $s(a_1, q) = 1$ means that we are looking for minimum elements from B (see section III-A) *s.t.* their union covers q: label similarity component equals one means that all nodes are matched and their labels are exactly similar, on the other hand, structure similarity component should be one, meaning that the query and the aggregate have similar structure. This reduces to resolving the NP-Hard set cover problem where the universe is the query q that we aim to cover with minimum elements from B. This ends the proof. ■

IV. Query Processing Algorithm

In this section, we present our query processing algorithm named LaSaS, which is a heuristic solution intended to solve the problem of aggregated graph search. We expose in details the proposed algorithm and its different steps.

LaSaS algorithm (described in algorithm 1) works in three distinct steps: first, the *Selection step* aims to select the most relevant fragments from the fragment set B based on the similarity metric s (see section III-B). Second, the *Aggregation step* combines the relevant fragments found by the *Selection* step to form the aggregate a. Third, the *Refinement step* enhances the quality of the aggregate by pruning irrelevant nodes and by mapping paths of a thresholded length to unmapped edges if any. Note that this threefold process is necessary for obtaining one aggregate, whereas in the general case of k aggregates, this process will repeat k times. In the following, we present each step in details.

1) Selection step: The first step consists in selecting the most relevant fragments, as many as necessary to cover the whole query q. The selection is based on successive iterations of two main substeps: *(i)* similarity checking and *(ii)* query updating until a stopping condition is verified. In the similarity checking, a rank is associated to each fragment f_i in B which is given by the similarity metric $s(f_i, q)$ (see section

Algorithm 1 *Label and Structure Similarity Aggregated Search*

Input: Fragments set B, Query graph q, number of expected aggregates k.
Output: Answer set A.

```
1.  i = 0;
2.  while i < k
3.      Select potential fragment from B that are similar to q;
4.      Add the selected fragments to set S;
5.      Aggregate all fragments in S to form an aggregate a_i;
6.      Refine a_i;
7.        Prune irrelevant nodes;
8.        If there are unmapped edges in a_i:
9.          Map paths to missing edges;
10.     A = A ∪ {a_i};
11.     i = i + 1;
12.     S = ∅;
13.   end while
14. return A;
```

III-B), then the fragment f_{max} having the maximal similarity is selected and we have $f_{max} = \underset{f_i}{argmax}\ s(f_i, q)$ where $f_i \in B$. The query is then updated according to the best-ranked fragment f_{max} such that the query part that has been covered by f_{max} is withdrawn according to several conditions which will be detailed in the following. When the query is updated, the selection process repeats: similarity checking and query update substeps are performed until one of these stopping conditions are verified: *(i)* $|V_q| < \epsilon$, meaning that almost all query nodes have been matched and/or *(ii)* the fragment set B is empty, given that a fragment is removed from B once chosen during the selection step.

Query Update. The foremost role of query update step is to ensure complementarity among selected fragments. This step updates the query by removing nodes that are covered by the selected fragment, thus enabling subsequent selections to choose fragments covering complementary parts of the query. The query update can be done in two distinct ways: using Maximum Common Subgraph (MCS), and using Weight Update (WU).

Query update using MCS. As aforementioned, the query is updated given the best fragment f_{max}. First, the maximum common subgraph (MCS) of q and f_{max} is computed, then the MCS is withdrawn from q while keeping the boundary nodes that belongs to the MCS as explained in the following.

Let q' denote the updated query q. We have:

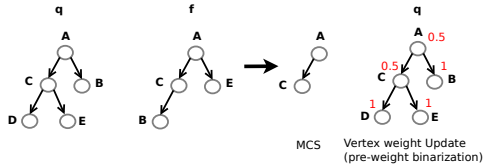
- $V_1(q') = \{v | v \in V_q \ \& \ v \notin MCS(q, f_{max})\}$
- $V_2(q') = \{u | (u, v) \in E_q \ \& \ v \notin MCS(q, f_{max})\}$

The vertex set of q' is $V_{q'} = V_1(q') \cup V_2(q')$, and the edge set is $E_{q'} = \{(u, v) \in V_{q'}^2 \mid (u, v) \in E_q\}$.

Query Update using Weight Update (QUWU). Computing the maximum common subgraph is known to be NP-complete by reduction from the maximum clique problem. Hence, it is cost prohibitive to use it in our method in such a repetitive operation as the query update. Alternatively, we perform a weight update on the query s.t. removable nodes on the query will be weighted by 0 and by 1 otherwise. In a nutshell, QUWU works in 2 steps: First, covered nodes will have their weights set to 0.5 as follows: $\forall (u, v) \in E_{f_{max}}$: if $(Q(u), Q(v)) \in E_q$ then $w(u) = w(v) = 0.5$, where $w(u)$ is the weight of vertex u. Figure 1 depicts the equivalent result to computing the MCS. Second, weight binarization is performed, *i.e.*, among covered nodes, removable nodes will be weighted by 0, otherwise by 1. QUWU process will not be further detailed due to lack of space, however, it is worth mentioning that it has a polynomial time complexity *w.r.t.* the number of query nodes.

Consequently, the stopping condition of the selection step would be that all vertices of q are weighted by 0, *i.e.* $W \leq \epsilon$, where $W = \sum_{v \in V_q} w(v)$.

Fig. 1. Vertex weight update vs. Maximum Common Subgraph.



Since QUWU does not reduce the query size as it does not withdraw vertices but update their weights, it is necessary to tweak the similarity function s (see equations 5 and 6) in order to consider only vertices that are weighted by 1, *i.e.*, not previously covered.

$$\Delta_L(f_i, q) = \frac{1}{n_i}. \sum_{\substack{v \in V_{f_i} \\ w(Q(v)) \neq 0}} J(\mu(v), \mu(Q(v))) \tag{5}$$

$$\Delta_D(f_i, q) = \frac{1}{n_i}. \sum_{\substack{u, v \in V_{f_i}, u < v \\ w(Q(v)) \neq 0}} \delta_t(d(u, v), d(Q(u), Q(v))) \tag{6}$$

2) Aggregation step: The fragments obtained upon successful completion of the selection step are stored in the solution set denoted by S. The aggregation step constructs an aggregate a_i from the solution set S as follows:

$$V_{a_i} = \bigcup_{f_i \in S} V_i \ \ and \ \ E_{a_i} = \bigcup_{f_i \in S} E_i$$

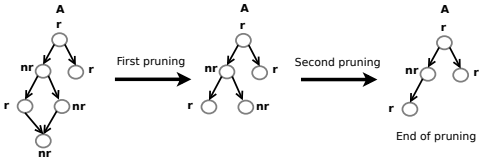
3) Refinement step: Refinement of aggregate a_i is achieved by *(i)* connecting the aggregate whenever it is disconnected, and *(ii)* by pruning irrelevant nodes from a_i.

Connecting the aggregate. Cases when a_i is disconnected may occur when selected fragments cover disjoint areas of query q leaving some query edges unmapped. Our algorithm maps these edges to paths by performing a path search in B in order to interconnect the connected components (*ccs*) in a_i. To this end, we search for a path between any two nodes belonging to different *ccs* of a_i in the graph G, where $G = \bigcup_{i=0}^{p} f_i$. For an optimal path finding task, we customized the Dijkstra algorithm by adding a constraint thresholding the path length, where only paths having a relatively small length are considered in order to speed up the search time, as long paths search will be abandoned once the threshold exceeded and, on the other hand, to preserve semantic relevance within a_i. Finally, the path search stops once a_i becomes connected.

Pruning. The aggregate a_i is further refined by withdrawing irrelevant nodes. To do so, we iteratively prune irrelevant leaves from a_i until no irrelevant leaf is left, where a leaf

is a vertex of degree 1. As depicted in figure 2, not all irrelevant nodes are pruned from A by the end of the pruning process, however, by limiting the pruning to irrelevant leaves, the connectivity of the aggregate is preserved as the removal of an irrelevant node with a degree>1 may disconnect the aggregate.

Fig. 2. Pruning process of the aggregate.



V. Experimental evaluation

This section shows the empirical evaluation of the proposed approach. We first describe the experimental set up along with the graph dataset and the evaluation methodology. Then we present the evaluation metrics. Finally, we present and discuss the obtained results.

A. Experimental set up

Graph Dataset. In order to evaluate our method, we used the real life dataset DBpedia Knowledge Base [23] that consists of RDF triples extracted from Wikipedia. We considered a total of 666043 triples from ten different entities.

Fragments generation. In order to constitute the set of fragments B, we partition the dataset used in such a way that no fragment is isomorphic to the query q. To avoid crossing edges between fragments, nodes belonging to the crossing edges are duplicated in the fragments involved.

Query generation. Two parameters are necessary to generate a query: the number of nodes denoted by n and the query diameter d which represents the largest distance between any two nodes. A query is generated by extracting a subgraph from the dataset and introducing some label and structure noise to it. The label noise is generated by randomly modifying some words in the original label while the structure noise is provided by randomly removing or adding some edges.

Evaluation metrics. The F1-measure is used as the main evaluation metric to show the effectiveness of our method. It combines the recall (R) and precision (P) where the recall shows the ratio of the correctly found node matches overall correct matches, and the precision is the ratio of correctly found matches overall found matches. F1-measure is as follows:

$$F1 = \frac{2}{(1/R + 1/P)}$$

In addition to he F1-measure, the runtime is also reported.

Methodology. Our experiments are based on the following guidelines: We create 4 query sets, and we generate 100 query under each set. First, we assess the influence of the structural noise on our method by fixing the label noise and varying the structural noise for all the 4 query sets. Similarly, to assess the label noise influence on our method, we fix the structural noise and vary the label noise on all the 4 query sets. We vary the ratio of the matching nodes in the query, *i.e.*, the query nodes that are surely present in the fragments set B, and see the repercussion on the performances of our method. Last but not least, we compare LaSaS to state-of-the-art tools: SAGA and BLINKS [24], a keyword graph search method.

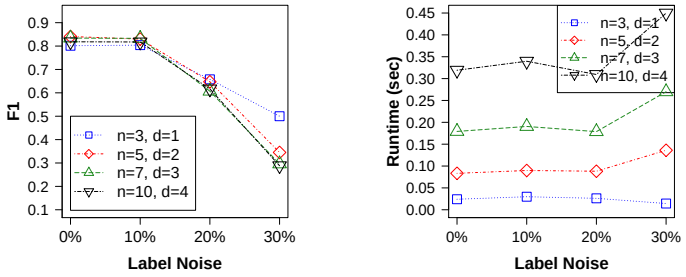
All algorithms have been implemented in C++, and all experiments were performed on a single machine, with Intel Xeon(R) CPU at 1.9GHz, and 16GB of main memory.

B. Experimental results

Label noise influence. We vary the label noise and report the F1-measure in figure 3. Having the structural noise fixed to 10%, F1-measure does not reach 1 when label noise equals 0%. Results show that LaSaS maintains the same performances for 0% to 10% of label noise, meaning that it efficiently discovers the correct matches despite the noise. However, when the label noise goes up to 30%, F1-measure drops considerably, which shows that LaSaS is label noise sensitive.

In figure 3, we show the runtime for all query sets while varying the label noise. Results show that for each query set, the runtime is even for different values of label noise, however, a slight increase of the runtime is noticed when the label noise reaches 30%, which is normal as the method spends additional time to look for other candidate matches. Altogether, increasing the label noise does not incur a considerable computational cost.

Fig. 3. F1-measure and Runtime (in seconds) reported upon four different query sets with 100 query within each set while varying the label noise, and the structural noise was set to 10%.



Structural noise influence. Figure 4 reports results for different values of structural noise while label noise was set to 10%. When no structural noise is added, the F1-measure does not attain 1 due to the label noise being set to 10%. Moreover, the F1-measure does not have an abrupt variation for all query sets (with the lower bound being 0.8 and the upper bound being 0.86), which translates the capacity of LaSaS to find the correct matches despite the added noise on the structure.

Figure 4 reports the runtime while we vary the structural noise, and results show that the runtime is almost steady for all query sets, that is to mention that the computational cost incurred by the added structural noise is negligible.

Ratio of query matching nodes. We refer to the ratio of matching nodes that are already in the fragments set B by Φ, and we investigate its influence on the performances of LaSaS. Figure 5 reports the F1-measure and shows that performances are optimal when the ratio of the matching nodes is equal to 100%, and it decreases gradually when the ratio of the matching nodes decreases. This shows the effectiveness of our method, as it finds the best results when they are available.

The reported runtime as shown in figure 5 increases with the ratio Φ, which means that the method takes more time to process as there are additional relevant answers in the B set.

LaSaS *vs.* SAGA and BLINKS. In Table I, we compare our method LaSaS to two state-of-the-art tools SAGA and BLINKS. Results in table I show that LaSaS outperforms both SAGA and BLINKS in effectiveness by achieving a greater F1-measure and in efficiency by finding results faster.

Fig. 4. F1-measure and Runtime (in seconds) reported upon four different query sets with 100 query within each set while varying the structural noise, and the label noise was set to 10%.

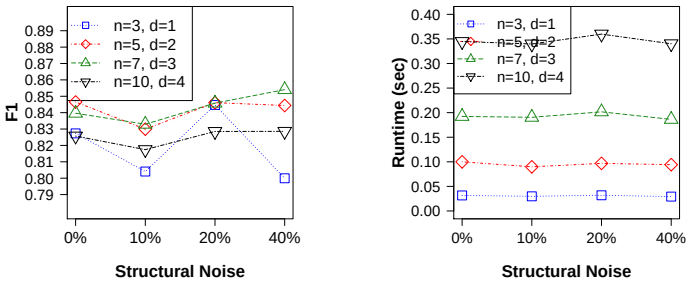
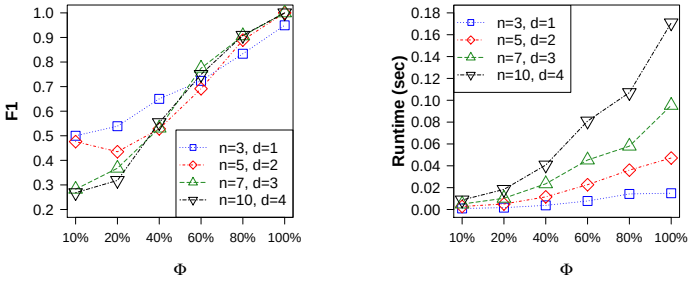


Fig. 5. F1-measure and Runtime (in seconds) reported upon four different query sets with 100 query within each set while varying the matching nodes ratio in the B set.



VI. Conclusion

In this paper, we discussed a novel framework for approximate graph matching based on aggregated search called Label and Structure Similarity Aggregated Search (LaSaS). The proposed approach enables an effective graph querying without any knowledge of the schema of the data graph. The framework joins ideas from aggregated search and graph matching to effectively find approximate matches for a query from a set of heterogeneous graphs. LaSaS is based on three key ideas: *(i)* aggregated search strategy in order to enrich the set of answers *(ii)* a lightweight graph similarity metric that takes into account both the nodes label and graph structure similarity to enable finding approximate matches *(iii)* a graph weight update that replaces the maximum common subgraph search task and reduces the complexity cost. Our method allows approximate matching by allowing slight label difference and by mapping edges to paths with a thresholded length. This feature makes our method unrestrictive and hence enable it to find more answer results compared to existing strict matching schemes like graph isomorphism. Empirical evaluation over the real life DBpedia graph [23] illustrates the effectiveness of our method over different parameter settings and corporates the stability of LaSaS. Moreover, the experimental results indicate that the proposed method outperforms the state-of-the-art related approaches by finding more precise matches. Future works will be conducted to build an index on the query and fragments to further accelerate the process of the selection step. We also plan to extend the empirical study to compare our method to additional matching tools on supplementary graph datasets.

Acknowledgements.

This work is partially funded by the French National Agency of Research project: Contextual and Aggregated Information Retrieval (ANR-14-CE23-0006).

TABLE I. Comparison results of LaSaS, SAGA and BLINKS: F1-measure in terms of nodes and graphs and runtime for finding the best match (top-1) over 100 queries of set 2 (n=5, d=2). Label noise and structure noise where both set to 10%.

	LaSaS	SAGA	BLINKS
F1-measure (nodes)	**0.93**	0.74	0.62
F1-measure (graphs)	**0.9**	0.7	0.58
Runtime (sec)	**0.15**	10.83	1.81

References

[1] P. Papadimitriou, A. Dasdan, and H. Garcia-Molina, "Web graph similarity for anomaly detection (poster)," ser. WWW '08.

[2] A. Schenker, M. Last, H. Bunke, and A. Kandel, "Classification of web documents using graph matching," *IJPRAI*, 2004.

[3] M. Aery and S. Chakravarthy, "emailsift: Email classification based on structure and content," ser. ICDM '05.

[4] H. Elghazel and M.-S. Hacid, "Aggregated search in graph databases: preliminary results," in *International Workshop on Graph-Based Representations in Pattern Recognition*, 2011.

[5] T.-H. Le, H. Elghazel, and M.-S. Hacid, "A relational-based approach for aggregated search in graph databases," in *DASFAA*, 2012.

[6] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, 2004.

[7] H. Shang, Y. Zhang, X. Lin, and J. X. Yu, "Taming verification hardness: An efficient algorithm for testing subgraph isomorphism," *VLDB*, 2008.

[8] A. Khan, N. Li, X. Yan, Z. Guan, S. Chakraborty, and S. Tao, "Neighborhood based fast graph search in large networks," in *Proc. of the 2011 ACM SIGMOD*.

[9] L. Zou, J. Mo, L. Chen, M. T. Özsu, and D. Zhao, "gstore: Answering sparql queries via subgraph matching," *Proc. VLDB Endow.*, 2011.

[10] J. R. Ullmann, "An algorithm for subgraph isomorphism," *J. ACM*, 1976.

[11] W. Fan, J. Li, S. Ma, N. Tang, Y. Wu, and Y. Wu, "Graph pattern matching: From intractable to polynomial time," *VLDB*, 2010.

[12] S. Ma, Y. Cao, W. Fan, J. Huai, and T. Wo, "Capturing topology in graph pattern matching," *Proc. VLDB Endow.*, 2011.

[13] W. Fan, J. Li, S. Ma, H. Wang, and Y. Wu, "Graph homomorphism revisited for graph matching," *Proc. VLDB Endow.*, vol. 3.

[14] Y. Tian, R. C. Mceachin, C. Santos, J. M. Patel *et al.*, "Saga: a subgraph matching tool for biological graphs," *Bioinformatics*, 2007.

[15] B. P. Kelley, B. Yuan, F. Lewitter, R. Sharan, B. R. Stockwell, and T. Ideker, "Pathblast: a tool for alignment of protein interaction networks," *Nucleic Acids Res*, 2004.

[16] Z. Liang, M. Xu, M. Teng, and L. Niu, "Netalign: A web-based tool for comparison of protein interaction networks," *Bioinformatics*, 2006.

[17] R. Singh, J. Xu, and B. Berger, "Pairwise global alignment of protein interaction networks by matching neighborhood topology," in *Proc. of RECOMB'11*.

[18] V. S. Cherukuri and K. S. Candan, "Propagation-vectors for trees (pvt): Concise yet effective summaries for hierarchical data and trees," in *Proc. of the LSDS-IR'08 ACM Workshop*.

[19] J. W. Kim and K. S. Candan, "Cp/cv: Concept similarity mining without frequency information from domain describing taxonomies," in *Proc. of CIKM'15*.

[20] J. R. Anderson, "A spreading activation theory of memory," *Journal of verbal learning and verbal behavior*, 1983.

[21] B. Gallagher, "Matching structure and semantics: A survey on graph-based pattern matching," *AAAI FS*, 2006.

[22] A. Kopliku, K. Pinel-Sauvagnat, and M. Boughanem, "Aggregated search: A new information retrieval paradigm," *ACM Comput. Surv.*, 2014.

[23] "http://dbpedia.org."

[24] H. He, H. Wang, J. Yang, and P. S. Yu, "Blinks: Ranked keyword searches on graphs," in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '07.

An Event Search Platform Using Machine Learning

Marcelo Aguiar Rodrigues
Polytechnic of Coimbra
ISEC
Coimbra, Portugal
a21180873@isec.pt

Rodrigo Rocha Silva
São Paulo State Technological College
CISUC
Mogi das Cruzes, Brasil
rodrigo.rsilva@fatec.sp.gov.br

Jorge Bernardino
Polytechnic of Coimbra - ISEC
CISUC
Coimbra, Portugal
jorge@isec.pt

***Abstract*—Currently, the evolution of technology allows to find which events occurs around us at any given location. Social networks are one of the reasons of this trend and new applications are emerging aiming at finding and disclosing events. This paper proposes a platform of event searching. In particular, we propose a new architecture that uses machine learning to classify events with tags. An experimental evaluation with different types of algorithms was done using Facebook as a source of dataset events.**

Keywords - events search; data mining; machine learning

I. INTRODUCTION

Nowadays, the quantity of digital information about what happens around us is dispersed in many applications. Usually, working with events, implies dealing with variables like date, location and time [1]. With the emergence of social networks, other types of relevant information should be considered, like a list of users that have an interest in the event or a list of users who will attend the event. In the literature, it is usually mentioned that users like sharing their stories, opinions, photos, and videos on social networks, creating a direct and social interaction between the participants on a certain event [2].

The events are a natural way to show an observable occurrence, grouping people, places, times, and activities [3]. Also, they might be considered as observable experiences that are often documented through photos and videos [4].

This paper presents a new idea of a platform for event searching. In particular, we propose a new architecture using machine learning to provide more accurate information according to the user interests. The main advantage of the platform is to bring a more personalized system where the user can find what s/he needs and get recommend events based on personalized tags that s/he follows.

Our main contributions are: a new approach for an event search platform using machine learning; integration of LODE ontology to structure event data and use it on classification; and classification tests with 101,121 events with 83.33% of classified events.

The remainder of this paper is organized as follows. Section II describes our event search platform and its architecture. Section III describes the process of organization data with the LODE ontology. Section IV describes the algorithms used and the experimental tests for events classification. Finally, Section V concludes the paper and presents future work.

II. THE EVENT SEARCH PLATFORM

Finding digital content related to events is challenging, requiring searching at different sources and sites [5] and sometimes, the data is ambiguous and incomplete.

A. The idea

The goal is to create an event search platform where every event can be classified with several tags. A good similarity is for example the Foursquare application [6]. Each place is associated to multiple tags, e.g., a restaurant can be associated to pasta, cocktails, pizza and, others, depending on their service type.

Our idea is to take advantage of these tags system and apply it on an event search platform. For example, a Bruce Springsteen concert [7] may be associated with tags like rock, hard rock or folk. Merging these two concepts (events and tags) can bring some advantages, such as:

- The platform can accommodate not only predefined events with selected tags, but all kind of events. For example, we can have one event related with music and one event related with a scheduled construction work on a specific street;
- Creation of customized lists according to the user's preferences;
- Creation of a more personalized search engine to return more accurate events to the user;
- Better interactivity with users, allowing them to create and classify events with tags. If a tag does not exist, the user can create the tag at the time of creating the event, allowing the system cover all type of events with the user input. As a business rule, each event should have at least one tag.

Machine learning is used for events classification to bring more improvements in the recommendation and search of events, as well as on the notification of events. Its main goal is to classify events obtained from APIs in several tags, but it can also help make the system more personalized to the user. For the platform, we can add a new feature like the suggestion of tags in the process of creating an event. For example, if a tag is followed by 1000 users, the suggestion of this tag at the time the event is created, can reach a larger number of people who might be interested in participating in it.

Yet, there is some concern about allowing users to create their events as well as classify them. This feature may lead to

DOI Reference Number: 10.18293/SEKE2017-159

inconsistent data and may have repercussions on the events classification. To solve this problem, we can use the recommendation system proposed in [8]: at the time of creating the event, the platform recommends a series of tags that can be used to classify the event depending on its data. If it is necessary to create a new tag, the submitted event must go through an approval process, to verify that the tags created are related with the event. This way, we think that it is possible to solve the issue of data inconsistency generated by the user.

B. *Proposed architecture*

Figure 1 shows the architecture of our platform and how its components communicate between them. Next, we will describe every component and its main function.

Client Applications are the applications that allow the user interact with the system and view the lists of events as well as create their own events. These applications will communicate with the server through the developed API, which is based on HTTPS (Hypertext Transfer Protocol Secure) protocol. The data sent is in JSON (JavaScript Object Notation) format and consists of event data.

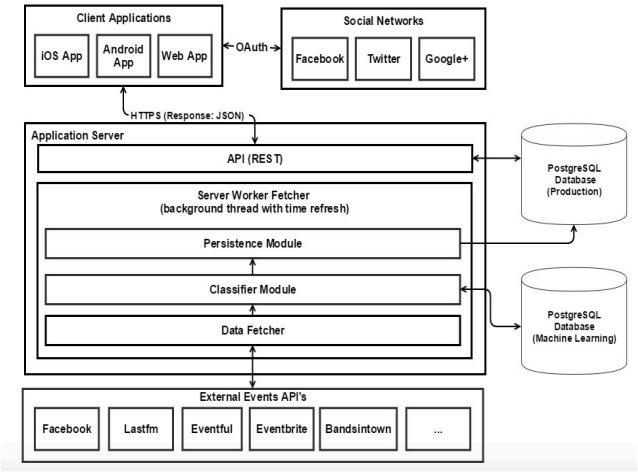


Figure 1. Proposed architecture

Application Server is the first responsible for supporting client applications, containing a RESTful API [9] to handle the requests about events. There is also a module for managing event classification (Server Worker Fetcher).

In order to save the data, the current architecture provides two databases. The PostgreSQL (Production) database will be the database that stores all event data already classified and used by the applications described above. The PostgreSQL (Machine Learning) is a copy of PostgreSQL (Production) database and will only be used as training base of the algorithm to classify the events coming from external APIs. This database will be updated periodically to improve event classification.

Server Worker Fetcher is a server worker whose main function is to get and classify events. It is divided into three modules: Data Fetcher module handles the communication between external APIs and the server to get events data; Classifier module handles the classification of event data through a machine learning algorithm. The PostgreSQL (Machine Learning) database provides the data for classification; Persistence Module stores event data already classified into the PostgreSQL (Production) database.

Finally, the external APIs are the APIs responsible to provide events.

III. EVENT DATA WITH LODE ONTOLOGY

One of the main contributions of this work is to show a different approach of an event search platform using machine learning. This section aims to present the process of integrating the LODE ontology to structure the event data and use them in the classification. A comparative study of several APIs was carried out, to understand which entities are similar between them.

The purpose of LODE ontology is to enable interoperable modelling of factual aspects to encapsulate the most useful properties to describe events [3]. The goal is to give answers about *"What is happening?"*, *"Where it is happening?"*, *"When it is happening?"*, *"Who is involved?"* [3], and organize this information in several properties, which are *Event*, *atPlace*, *atTime* and, *involved*.

Events often need a response from the user. This response is called R.S.V.P, which means "Répondez S'ill Vous Plaît" in French. This data permits to know if whether users will attend the event. This status is represented in several users counts that can be subdivided into the following categories:

- Attending guests: represents the guests that will attend the event;
- Declined guests: represents the guests that won't be attending the event;
- Interested guests: represents the guests that have interest in the event but don't know if they will be attending;
- No reply guests: guests who didn't reply to the invite;
- Maybe guests: guests that maybe will attend the event.

The final attributes of our dataset can be seen on Table I:

TABLE I. *Attributes of our dataset*

Properties	Attributes
atPlace	venue_latitude
	venue_longitude
atTime	event_start_hour
	event_end_hour
	event_start_day_of_month
	event_end_day_of_month
inSpace	venue_id
involved	artist_id
social	event_attending_count
	event_declined_count
	event_interested_count
	event_noreply_count
	event_maybe_count

IV. EXPERIMENTAL EVALUATION

In the experimental evaluation, we intend to find the best classification result in order to validate events classification for only one tag.

A. Algorithms

We use the following algorithms provided by *Weka,* corresponding to different classification categories: Decision Trees, was chosen the Random Forest [10], for the lazy classifiers, the K-Nearest Neighbors [11] was chosen, whose implementation in Weka is named IBk and, for function classifiers, Sequential Minimal Optimization (SMO) [12] was chosen, an algorithm for training support vector machines.

B. Classification results and discussion

In order to perform these experiments, it was necessary to create two datasets. The first dataset has about 1,121 events sourced from Facebook. The second dataset is a generated dataset with about 100,000 events.

The tests performed in this work are evaluated using the correctly classified instances. The 10-fold cross validation test mode was used, which means that 90% of the data is used for training and 10% for testing in each fold test.

The first test aimed to get the classification results for both datasets, to understand the first results, without changing the data as well as the algorithms. Table II shows the difference between the results obtained for the dataset with Facebook events in relation to the randomly dataset.

TABLE II. *Results of the first classification test*

	% of correctly classified instances	
Algorithms	***Facebook Dataset***	***Randomly Dataset***
IBk	50.02%	100%
SMO	46.67%	100%
Random Forest	70.28%	100%

For the IBk and SMO algorithms, the difference is around 50% and for the Random Forest algorithm the difference is around 30%. These differences are related to some missing values in the Facebook dataset. Since the dataset is composed by numeric data, the APIs do not always return all data to the attributes, leading to missing values. These same values are represented as zero, which on our view, affects the classification of events. There are three approaches to lead with missing values: mark, impute and remove missing values.

The technique to mark missing values aims to change the missing data that will be represented as "?". Yet, instances with missing values do not have to be removed and we can replace the missing values with other values with the mean of the numerical distribution. In order to have also missing data on our generated dataset, we created another one and we did the same tests for this new dataset. The results for these two techniques described above can be seen on Table III.

Comparing the results of Table II with Table III for the generated dataset, we can conclude that adding values missing also made the results worse. It is clear that both approaches cannot be taken into account in the classification process.

The last approach to deal with missing values is to remove events that contain one or more attributes with missing data. Considering the results of the previous Table II and III, we chose Feature Selection to understand which attributes are the most useful or relevant to our scenario. This is important because the number of attributes used can make the work of the classifier more difficult, making it slower and even diminishing accuracy.

TABLE III. *Classification results for mark and impute missing values techniques*

Algorithms	**Technique**	**% of correctly classified instances**	
		Facebook Dataset	***Randomly Dataset***
IBk	Mark Missing Values	41.60%	62.09%
	Impute Missing Values	48.12%	68.88%
SMO	Mark Missing Values	43.86%	77.96%
	Impute Missing Values	43.89%	76.33%
Random Forest	Mark Missing Values	61.54%	70.45%
	Impute Missing Values	68.89%	79.44%

Feature selection method aids to create an accurate predictive model. They help choose features that will give good or better accuracy whilst requiring less data [13]. They can be used to identify and remove irrelevant or redundant attributes from data that do not contribute to the accuracy of a predictive model or can decrease the accuracy.

Many feature selection techniques are supported in Weka. We choose the Information Gain Based Feature Selection, a popular technique to calculate the information gain based on the entropy concept. It is used as a measure of feature relevance in filter strategies that evaluate a feature individually [14]. We can calculate the information gain for each attribute for the output variable. Entry values vary from 0 (no information) to 1 (maximum information). Those attributes with more information will have a higher information gain than the others. Since the Facebook dataset represents the actual data of our platform, we only applied this technique on this dataset to understand the most relevant attributes. Table IV only shows the attributes that have a contribution for our case.

TABLE IV. *Attributes contribution gain results*

Attributes	**Information Gain**
artist_id	0.8864
event_start_hour	0.4246
event_end_day_of_month	0.4246
venue_longitude	0.3639
event_maybe_count	0.1003
event_interested_count	0.0600
event_attending_count	0.0423
venue_id	0.0403

We used an arbitrary cut-off of 0, which means that the attributes with this value were removed from the dataset. We proceeded again to the classification tests with the changes made on the dataset. The results can be seen on Table V.

Table V shows a great improvement comparing with results of Table II. Random Forest increased 13.05%, IBk increased 27.02%, and SMO increased 23.07%. This feature selection

showed that we have a lot of irrelevant attributes making the classifiers slower and even in some cases diminishing its accuracy.

TABLE V. *Classification results after apply the Information Gain Feature Selection*

Algorithms	% of correctly classified instances
IBk	77.74%
SMO	69.74%
Random Forest	83.33%

In conclusion, given the large difference in the results between Table II and Table III, compared with Table V, it is possible to verify that one of the problems of our dataset and the unsatisfactory results of the first two tests are related with the missing data. However, with the use of Information Gain feature selection technique, when classifying with only the most relevant attributes of our dataset, even with missing data, the results have risen considerably. In addition, within the relevant attributes it is possible to observe that 3 attributes are related with R.S.V.P, confirming that they bring relevant data in the classification of events.

In a first phase, feature selection needs to be applied since it allows to remove immediately the redundancy and irrelevance of some attributes. Even for a large database with 100,000 events, if we don't have missing data, the results are very good, as shown in Table II, but if we add missing data the results are worst. In this case, techniques of remove missing values should be applied, to understand the impact of these missing data in the dataset.

V. CONCLUSIONS AND FUTURE WORK

We propose an event search platform with a new architecture using machine learning. The use of machine learning aims to classify events in a specific tag. Our idea takes advantage of a tags system to agglomerate, not only predefined events on some categories, but all kind of events. Other advantages of our proposal, are to create customized data according to the user's preferences and a better interactivity between the users and events.

The LODE ontology was used to organize the data obtained from external APIs and was made an experimental study to find the best classification result and algorithm to validate the addition of machine learning on the architecture proposed. For performing these experiments, it was necessary to create two datasets: the first dataset has about 1,121 events sourced from Facebook and the second dataset is a generated dataset with about 100,000 events.

From the three algorithms used (Random Forest, IBk, and SMO), the first results weren't satisfactory for the Facebook dataset. The best result was 70.28% for the Random Forest algorithm. But, for the generated dataset, the results were good, reaching 100% of classified instances.

From the experimental tests, it was verified that sometimes the APIs return missing values which leads to a poor classification of the algorithms. Using the feature selection technique, we came to the conclusion that certain attributes of the Facebook dataset were irrelevant. After being eliminated, Random Forest obtained the best classification result, reaching 83.33% of classified instances. Comparing the results of the generated dataset in the beginning of tests with this result, it is possible to conclude that our training data can't have missing values because, the algorithms performance is worst

Although the classification result (83.33%) was good, there are open issues that we will be performed as future work. The experimental dataset has a small event base, so it is necessary to have more events to confirm the results obtained in these tests. With more events, other techniques of feature selection, such as, learner feature selection or correlation feature selection, must be considered, to understand the data generated in the dataset, to find a pattern that allows obtaining the best percentage for the classification of events.

All these results prove that the proposed platform is viable. Yet, allowing users to create and sort their events, the ambiguity and inconsistency of the data may be a problem in the future. Despite the proposals presented in this paper to solve the problem, they must be validated. Also, it is also necessary to find a solution to merge the data coming from external APIs, since each API has its own data structure. These issues lead us to other relevant issues about the performance, such as: the time that takes to build our training base and prepare the data for classifications, the performance of a classification procedure and the combination of data among the external APIs.

REFERENCES

[1] Costa-Dasilva, Ignacio, J., Gómez-Rodríguez, A., González-Moreno, J.C. and Ramos-Valcárcel, D. (2015) 'A located and user personalized event's dissemination platform', Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology, 28(1), pp. 71–81.

[2] Baruah, T.D. (2012), "Effectiveness of Social Media as a tool of communication and its potential for technology enabled connections: A micro-level study", International Journal of Scientific and Research Publications, Volume 2, Issue 5.

[3] Shaw, R., Troncy, R. and Hardman, L. (2009) 'LODE: Linking open descriptions of events', in Lecture Notes in Computer Science. Springer Nature, pp. 153–167.

[4] Troncy, R., Fialho, A., EURECOM, Hardman, L. and Saathoff, C. (2010) 'Experiencing events through user-generated media'

[5] Girolami, M., Chessa, S. and Caruso, A. (2015) 'On service discovery in mobile social networks', Computer Networks: The International Journal of Computer and Telecommunications Networking, 88(C), pp. 51–71.

[6] Made, L. and SFfourSq (2016) Food, Nightlife, entertainment. Available at: https://foursquare.com/ (Accessed: 25 November 2016).

[7] Springsteen, B. (no date) Official Website. Available at: http:// brucespringsteen.net (Accessed: 22 November 2016).

[8] Ricci, F., Rokach, L. and Shapira, B. (eds.) (2015) Recommender systems handbook. Springer Nature.

[9] Garriga, M., Mateos, C., Flores, A., Cechich, A. and Zunino, A. (2016) 'RESTful service composition at a glance: A survey', Journal of Network and Computer Applications, 60, pp. 32–53.

[10] Breiman, L. (2001) Machine Learning, 45(1), pp. 5–32.

[11] Aha, D. W., Kibler, D. and Albert, M. K. (1991) 'Instance-based learning algorithms', Machine Learning, 6(1), pp. 37–66.

[12] Cohen, (1995), "Fast Effective Rule Induction," In Proceedings of the Twelfth International Conference on Machine Learning.

[13] Guyon, I. and Elisseeff, A. (2003) 'An introduction to variable and feature selection', The Journal of Machine Learning Research, 3, pp. 1157–1182.

[14] Yang, Y. and Pedersen, J. O. A Comparative Study on Feature Selection in Text Categorization. In Proceedings of the 14th International Conference on Machine Learning. San Francisco, CA, USA, pp. 412–420, 1997

The effects of classifiers diversity on the accuracy of stacking

Mariele Lanes
Centro de Ciências Computacionais
Universidade Federal do Rio Grande
Rio Grande, Brazil
Email: mariele.lanes@furg.br

Eduardo N. Borges
Centro de Ciências Computacionais
Universidade Federal do Rio Grande
Rio Grande, Brazil
Email: eduardoborges@furg.br

Renata Galante
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Porto Alegre, Brazil
Email:galante@inf.ufrgs.br

***Abstract*—In recent years several data classification techniques have been proposed. However, it is not a trivial task to choose the most appropriate classifier for deal with a particular problem and set it up properly. In addition, there is no optimal algorithm to solve all prediction problems. In order to improve the result of the classification process, the stacking strategy combines the knowledge acquired by individual learning algorithms aiming to discover new patterns not yet identified. Stacking combines the outputs of base classifiers, induced by several learning algorithms using the same dataset, by means of a meta-classifier. The main goal of this paper is to evaluate the effects of classifier diversity on the accuracy of stacking. We have performed a lot of experiments which results show the impact of multiple diversity measures on the gain of stacking, considering many real datasets extracted from UCI machine learning repository and three synthetic two-dimensional datasets. The results revealed connections between some measures and the gain of stacking, but they imply a weak or moderate relationship that suggest predicting the improvement on the best base classifier accuracy using diversity measures is inappropriate.**

I. Introduction

Classification is the most usual task among data mining tasks. It is based on the discovery of prediction rules that aid in making decisions. Generally, this task is used when there are large amounts of records in a database, which have several fields, and it is necessary to extract from this base some relevant knowledge with predictive capacity.

Classification algorithms can be organized into different types according to the technical features they use in learning. Each type is best suited for a particular dataset. Although some classifiers individually provide solutions which are considered effective, the experimental evaluation performed by [1] shows a drop in the quality when there are large sets of patterns and/or a significant number of incomplete data samples or irrelevant features. That is, such classifiers may not effectively and/or efficiently recognize patterns in complex problems.

Classifiers that implement different algorithms potentially provide additional information on the patterns to be classified. The combination of the outputs of a set of different classifiers aims to get a more precise classification, i.e. to reach a greater accuracy. In this context, stacking [2] is a widely used method for combining multiple classifiers generated from different learning algorithms applied on the same dataset. It is also known in the literature as stacked generalization [3], [4].

DOI reference number: 10.18293/SEKE2017-016

In the stacking method the choice of base algorithms is very important. It is desirable that there be different solutions to the problem to be solved, i.e., it is important to obtain a diversity among the results found by these algorithms. According to [5], the performance of the stacking method strongly depends on the accuracy and diversity of classifiers results. To verify this diversity, there are several measures based on the agreement and/or disagreement of the classifiers [6].

Therefore, the use of base algorithms with different particulars is ideal, since the patterns learned tend not to be the same. Thus, even low accuracy classifiers combined can generate a strong classifier, providing gain for stacking. Otherwise, when several classifiers agree on the vast majority of responses (no diversity), the combination will possibly have the same result, with no improvement in the stacking quality.

The purpose of this paper is to evaluate the effects of classifier diversity on the accuracy of stacking. The experiments we have performed show the relationship between multiple diversity measures and the gain of stacking, considering a lot of real datasets extracted from UCI machine learning repository and three other synthetic two-dimensional datasets used for visual inspection.

II. Related work

Stacking method combines multiple base classifiers trained by using different learning algorithms L on a single dataset *S*, by means of a meta-classifier [7], [8]. Each training sample $s_j = (X_j, y_j)$ is a pair composed by an array of features X_j and the class label y_j.

The process can be described in two distinct levels. The first level-0 defines a set of N base classifiers, where $C_i = L_i(S)|1 \leq i \leq N$. Level-0 classifiers are trained and tested using the cross-validation or leave-one-out procedure. The output dataset D used for training the meta-classier is composed by examples $((y_j^1, \ldots, y_j^i), y_j)$, i.e. a vector of predictions for each base classifier $y_j^i = C_i(X_j)$ and the same original class label y_j [3]. In the second level-1, the meta-classifier combines base classifiers outputs from D into a final prediction y_j^f. The stacking pseudocode can be seen in Algorithm 1.

Algorithm 1: Combining classifiers with stacking

Input: training samples $s_j \in S$
Output: final predictions y_j^f

```
begin
    Select N learning algorithms (L_1, L_2, ..., L_N);
    for i = 1, 2, ..., N do
        Train C_i = L_i(S) using cross-validation;
        y_j^i = C_i(X_j);
    end
    Make up a new dataset D combining all predictions y_j^i;
    Train M = L(D) using cross-validation;
    y_j^f = M(D);
end
return y_j^f
```

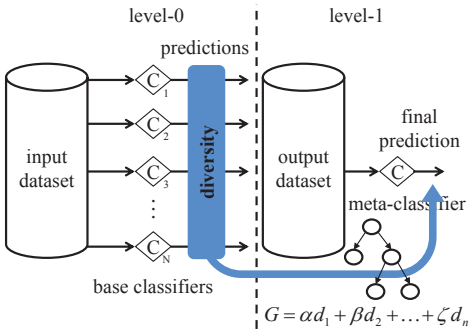


Fig. 1. The proposed method for analyzing the effects of classifiers diversity on the accuracy of stacking.

Several approaches have proposed the use of stacking to increase the classification quality in recent years [9], [10], [11], [12], [13]. Considering the covered related work, the classification techniques most used at level-0 are those based on decision trees, artificial neural networks and Bayes' theorem, which are usually combined by a function-based classifier. The majority of approaches stacks the predicted labels or the confidence of these predictions to compose the feature vector of level-1.

The diversity of predictions y_j^i is a key issue in the combination of classifiers. Reference [6] defines several measures of diversity and relate them to the quality of classification system. Experiment results revealed the used measures were strongly correlated between themselves and the possible inadequacy of the diversity measures for predicting the improvement on the best individual accuracy. Although there are proven connections between diversity and accuracy in some special cases, the results raised some doubts about the usefulness of diversity measures in constructing classifier sets in real-life pattern recognition problems. References [14], [15] also conclude that diversity measures can hardly be used as selection criteria for building ensembles. However, other authors report success using diversity to detect noise [16] and to generate more precise classification systems [17], [18], [19], [20].

III. Proposed Method

The proposed method for analyzing the effects of classifiers diversity on the accuracy of stacking are graphically represented in Figure 1. For each analyzed dataset, different learning algorithms are used to train multiple base classifiers. The predictions returned by these classifiers are evaluated and used to perform several measures of diversity [6]: double-fault df, Disagreement Dis, Q statistic, Correlation coefficient ρ, Kohavi-Wolpert variance KW, Interrater agreement k and Entropy E. These measures check whether and how the classifiers agree or disagree on the predicted class label. At level-1, classifiers predictions for each original instance are used to compose a new dataset that is submitted to another algorithm for training the meta-classifier. Final prediction is determined from the combination of knowledge learned by the base classifiers.

The feature vectors for each training set are made up of all data fields and the class label. The following algorithms are used at level-0 of the stacking method: Multilayer Perceptron (MLP) [21], a variation of Support Vector Machine (SVM) [22] called SMO [23], Nave Bayes (NB) [24], RIPPER [25], C4.5 [26] and Random Forest (RF) [27]. This choice was motivated mainly because the algorithms are quite heterogeneous, since they are based on distinct particulars. The meta-classifier is trained using any of these classification algorithms combining the knowledge learned by the base classifiers, and it is finally used to get a final prediction. The gain of stacking G is computed by the ratio between accuracy values achieved by the meta-classifier and by the best base classifier.

The effects of diversity on stacking can be analyzed by observing the relationship between diversity measures values and the gain of stacking for multiple datasets. It is expected that the most diverse sets of classifiers will contribute to the quality of stacking. This relationship is quantified using a linear regression function and a regression model tree induced by the algorithm M5 [28] with the gain of stacking G as the target field. The vector of features is composed by the diversity measures previously computed $(d_1, d_2, \ldots, d_n)$. The regression models show how much each measure pitches in with the gain of stacking.

IV. Experimental Evaluation

We have used 54 real classification datasets extracted from UCI[1]. The full list is available in the research group website[2]. The chosen datasets cover several areas of knowledge: business, computer, financial, game, life, physical and social. Many of them were widely cited in the scientific literature and they have sundry objectives. The field data types can be integer, real or categorical. The amount of instances ranges from 187 to 12,960. The number of fields and class labels varies from 4 to 216 and from 2 to 48 respectively. These datasets were deposited in the UCI repository from the year 1987 to 2015.

Furthermore, have used three two-dimensional synthetic datasets with very different spatial distributions that allow us to interpret results and algorithms behavior by visual inspection

[1] archive.ics.uci.edu/ml
[2] ginfo.c3.furg.br

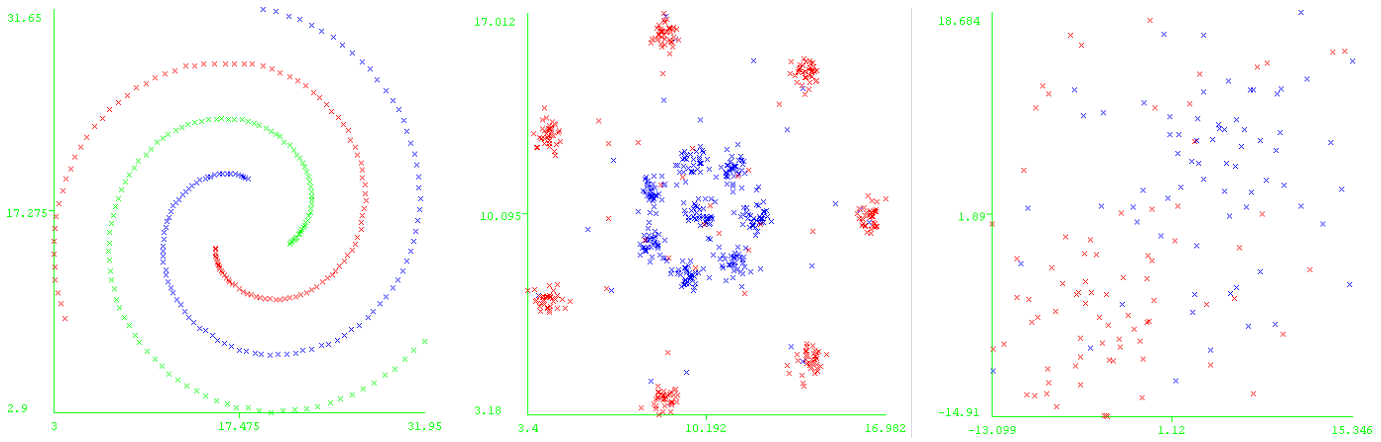


Fig. 2. Datasets Spiral, R15 and D13.

(see Figure 2). Spiral [29] consists of three concentric spirals. R15 [30] is composed by 15 similar Gaussian distributions. We reduced it to 2 classes and included little noise by changing the label of some instances manually selected. D13 [31] presents two Gaussians randomly parameterized and with a lot of noise (half of the instances randomly selected). The number of instances varies from 150 to 600.

The experiments were performed using the data mining tool Weka [32]. The algorithms were parameterized to improve the accuracy or using default values.

A. Diversity and stacking results

Experimental results are summarized in Table I and II that show for each dataset the following information: the computed diversity measures double fault df, disagreement Dis, statistic Q, correlation coefficient ρ, interrater agreement k, Kohavi-Wolpert variance KW and entropy E; the algorithm used to learn the best base classifier (L_0) and its accuracy in percentage (A_{L_0}); the algorithm used to learn the best meta-classifier (L_1) and its accuracy in percentage (A_{L_1}); and the gain of stacking (G), used to sort the results, also in percentage. Values of df, Dis, Q and ρ are averages of the computed values for each pair of base classifiers.

Table I presents experiments with real datasets that reached the worst and best values of G, i.e. we have omitted the results for any dataset where the gain of stacking ranges between -1 and 1%. Table II covers the synthetic datasets.

Observing Table I, we notice that stacking worked well only for 8 out of 54 datasets, where the gain in accuracy ranged from 1.2 to 5.1% (lines 1-8). The best gain of stacking was reached by Balance Scale dataset, in an already accurate result (90.7%) which is difficult to improve. The most frequent algorithm that reaches the best accuracy for level-0 was MLP ranging $26.6 \leq A_{L_0} \leq 90.7$, followed by RF with $84.8 \leq A_{L_0} \leq 92.9$. At level-1, the best meta-classifiers were trained with SMO $(26.9 \leq A_{L_1} \leq 94.9)$ and RF $(83.7 \leq A_{L_1} \leq 95.4)$.

However, stacking decreased the classification quality for some datasets (lines 9-17) reaching in the worst case $G = -9.4\%$. The most frequent algorithms with best accuracy were RF (L_0) and SMO (L_1). For some datasets, more than one classifier used at level-1 returned the same result. For instance, SMO and MLP reaches equal values $(A_{L_1} = 78.8\%)$ for Leaf dataset (line 9).

We have considered good values of diversity those that were sufficiently larger or smaller than the average for all 54 datasets, taking into account whether the measure is directly or inversely proportional to the diversity among the classifiers. These good values are in bold. A general analysis of them indicates that there is more classification diversity for the experiments in which there was gain of stacking (lines 1-8) than for those in which there was loss of quality (lines 9-17).

Abalone dataset (line 8) had the best value of double fault $(df = 0.09)$ due to the low accuracy $(A_{L_0} = 26.6\%)$ presented by the base classifiers. Many of them fail together because these is a multi-classification problem involving 28 distinct class labels. For datasets Connect. Bench (S,M vs. R), Statlog (Vehicle Silh.) and Diabetic Retinopat. Debrec. (lines 2-4) df values were less significant, compared to Abalone, but good in relation to the average. These df values are related to the better accuracy presented by the base classifiers and to the lower number of classes. We observe that for these datasets, all the measures of diversity return good values, collaborating with the hypothesis that the greater the diversity, the greater the stacking accuracy.

However, the experiment involving Low Resolution Spectrometer dataset (line 16) revealed the opposed behavior where the gain of stacking was negative $(G = -5.6\%)$, i.e. the quality of classification decreased considerably, even with high values for all measures of diversity. These high values are returned because there are 531 instances distributed in 48 classes, making even hard the agreement of many classifiers. Balance Scale dataset (line 1) is another counterexample in which there was no diversity among classifiers, but the stacking has reached the best G among all the performed experiments. Furthermore, diversity was considered non-existent or low in 33 out of 37 datasets where the gain of stacking ranged from -1 to 1%.

TABLE I
DIVERSITY MEASURES AND STACKING RESULTS FOR REAL DATASETS.

	Dataset	$df \downarrow$	$Dis \uparrow$	$Q \downarrow$	$\rho \downarrow$	$k \downarrow$	$KW \uparrow$	$E \uparrow$	L_0	A_{L_0}	L_1	A_{L_1}	G
1	Balance Scale	0.78	0.13	0.87	0.50	0.48	0.06	0.17	MLP	90.7	RF	95.4	5.1
2	Connect. Bench (S,M vs. R)	**0.62**	**0.27**	**0.58**	**0.28**	**0.26**	**0.11**	**0.35**	MLP	82.2	Jrip	85.6	4.1
3	Statlog (Vehicle Silh.)	**0.55**	**0.30**	**0.64**	**0.32**	**0.28**	**0.13**	**0.37**	MLP	81.7	RF	83.7	2.5
4	Diabetic Retinopat. Debrec.	**0.49**	**0.32**	**0.56**	**0.29**	**0.29**	**0.13**	**0.41**	MLP	72.0	SMO	73.8	2.4
5	Ionosphere	0.83	0.12	0.83	0.41	0.38	0.05	0.13	RF	92.9	SMO	94.9	2.2
6	Contrac. Method Choice	**0.38**	**0.29**	**0.71**	0.42	0.42	**0.12**	**0.36**	MLP	54.2	SMO	55.3	2.0
7	Vertebral Column	0.72	0.19	0.74	0.39	0.38	0.08	0.23	RF	84.8	RF	86.1	1.5
8	Abalone	**0.09**	**0.28**	**0.47**	**0.21**	**0.21**	**0.12**	**0.36**	MLP	26.6	SMO	26.9	1.2
9	Leaf	**0.52**	**0.30**	**0.70**	0.37	**0.33**	**0.12**	**0.37**	MLP	79.7	SMO$^{\#}$	78.8	-1.1
10	Glass Identification	**0.49**	**0.32**	**0.61**	**0.33**	**0.30**	**0.13**	**0.40**	RF	79.9	RF	79.0	-1.2
11	Credit Approval	0.77	0.14	0.85	0.50	0.48	0.06	0.17	RF	86.7	NB	85.7	-1.2
12	Ecoli	0.79	0.11	0.92	0.57	0.57	0.05	0.14	RF	87.2	RF	86.0	-1.4
13	Solar Flare	**0.62**	0.15	0.91	0.64	0.64	0.06	0.18	J48	72.1	SMO	70.9	-1.7
14	Dresses Attribute Sales	**0.46**	**0.26**	0.77	0.47	0.47	**0.11**	**0.32**	JRip	63.0	NB	60.2	-4.4
15	Audiology (Std.)	0.72	0.13	0.94	0.64	0.63	0.05	0.16	MLP	83.2	SMO	79.2	-4.8
16	Low Resolution Spectrom.	**0.18**	**0.36**	**0.47**	**0.25**	**0.23**	**0.15**	**0.46**	RF	54.0	SMO	51.0	-5.6
17	Primary Tumor	**0.33**	0.19	0.89	0.61	0.60	0.08	**0.25**	NB	50.1	RF	45.4	-9.4
	Average (all 54 datasets)	0.74	0.15	0.76	0.41	0.39	0.06	0.19					

$^{\#}$ MLP reaches equal results

Table II shows diversity and stacking results for synthetic datasets. Using Weka's default values to parametrize the classification algorithms (lines 1-3), R15 and Spiral present gain of stacking, but only classifiers applied to Spiral dataset were high diverse (values in bold). Manually configuring the parameters to reach best accuracy (lines 4-6), stacking decreased the classification quality for all 3 datasets, even with high diversity among classifiers.

To check whether the gain of stacking are in fact statistically significant, we performed a paired Student's T-test [33] comparing the accuracy values of the best meta-classifier (A_{L_1}) with those of the best base classifier (A_{L_0}). We used the T-test because it evaluates whether the means of two normal distributions of values are statistically different, showing good results even when the distributions are not perfectly normal.

Table III shows the gain of stacking and the values of 1-tailed p for each dataset where $p < 0.05$, i.e. when the stacking was statistically superior or inferior than the best base classifier. The number of observations (Obs.) was set to the amount of instances. The statistical difference occurred in 18 of the 54 experiments considering real datasets and only in two involving the synthetic ones. The gain of stacking for the other datasets was considered a technical tie. Datasets in which there was high diversity of the classifiers are highlighted in bold.

Experiments including datasets Connect. Bench (S,M vs. R), Abalone and Turkiye Student Eval. present good values for all diversity measures. Although the diversity measures applied to Spiral dataset have also returned great values, regardless of the parameter configuration, the gain of stacking was not considered significant because the best classifier achieves accuracy equal to 98.7%, which is very difficult to improve. For most experiments there seems to be no relationship between the classifiers diversity and the significant gain in stacking accuracy.

B. Diversity effects

We used regression models to quantify the effects of diversity on the gain of stacking applied to real datasets. We varied the learning algorithm (linear or M5) and the training set: 17 datasets with best and worst (b/w) G, where $-1 \geq G \geq 1$, present in Table I, 18 real datasets where the gain of stacking passed T-test present in Table III and considering all 54 real datasets. The models were evaluated using correlation coefficient and root relative squared error (RRSE).

For the training set composed by the 17 datasets with best and worst values of G, the minimum number of instances to allow at a leaf node in M5 algorithm ranged from 2 to 4, however the result was the same tree with only one node containing the model described by Equation 1. We notice that df had a positive effect on the gain but the influence of ρ was negative. Other diversity measures were irrelevant in estimating the gain. The correlation with the gain of stacking was 0.5243 and the RRSE was 79.67%.

$$G = 0.1278\ df - 0.2189\ \rho + 0.0168 \tag{1}$$

Considering only the 18 datasets that passed T-test, gain of stacking was affected only by ρ (Equation 2), which correlation was 0.3532 and RRSE equal to 99.89%.

$$G = -0.0847\ \rho + 0.0362 \tag{2}$$

Equation 3 shows the linear model trained with all 54 datasets. For this model, only df and KW had effect on the gain of stacking. ρ and other measures were not used. Correlation and RRSE was 0.4081 and 91.58% respectively.

$$G = 0.0971\ df + 0.3757\ KW - 0.0957 \tag{3}$$

TABLE II
DIVERSITY MEASURES AND STACKING RESULTS FOR SYNTHETIC DATASETS.

	Datasets	Parameters	$df \downarrow$	$Dis \uparrow$	$Q \downarrow$	$\rho \downarrow$	$k \downarrow$	$KW \uparrow$	$E \uparrow$	L_0	A_{L_0}	L_1	A_{L_1}	G
1	Spiral		**0.44**	**0.43**	**0.52**	**0.21**	**0.05**	**0.18**	**0.63**	RF	98.7	RF$^{\#}$	99.3	0.6
2	R15	default	0.76	0.19	**0.66**	0.48	**0.23**	0.08	0.21	J48*	93.8	SMO	94.5	0.7
3	D13		0.71	0.12	0.94	0.67	0.67	0.05	0.15	JRip	79.2	MLP	78.5	-0.8
4	Spiral'		0.75	**0.23**	**0.47**	**0.11**	**0.02**	**0.10**	**0.26**	RF	99.4	SMO$^{\#}$	99.0	-0.3
5	R15'	manually configured	0.93	0.02	1.00	0.87	0.87	0.01	0.02	J48	94.0	JRip^{*+}	93.7	-0.4
6	D13'		0.72	0.10	0.95	0.71	0.70	0.04	0.13	JRip^{+}	79.2	SMO	76.5	-3.4

$^{\#}$ MLP reaches equal results * NB reaches equal results $^{+}$ J48 reaches equal results

TABLE III
STATISTICAL TEST APPLIED TO THE GAIN OF STACKING.

	Set	Dataset	Obs.	p	G
1	real	**Connect. Bench (S,Mvs.R)**	208	0.008	4.1
2		Ionosphere	351	0.008	2.2
3		Vertebral Column	310	0.045	1.5
4		**Abalone**	4177	0.001	1.2
5		Mammographic Mass	961	0.025	0.6
6		Soybean	683	0.045	0.6
7		Tic-Tac-Toe Endgame	958	0.025	0.5
8		QSAR biodegradation	1055	0.045	0.4
9		Wilt	4839	0.001	0.3
10		Spambase	4601	0.014	0.1
11		Chess (KR vs. K-P)	3196	0.046	0.1
12		**Turkiye Student Eval.**	5819	0.025	0.1
13		Phishing Websites	11,055	0.003	0.1
14		Credit Approval	690	0.008	-1.2
15		Ecoli	366	0.045	-1.4
16		Solar Flare	323	0.045	-1.7
17		Dresses Attribute Sales	501	0.001	-4.4
18		Audiology	226	0.003	-4.8
19	synthetic	R15	600	0.005	0.7
20		D13'	150	0.045	-3.4

Table IV summarizes the best results comparing the evaluation of linear regression and model trees. We notice that the correlation between the diversity measures and the gain of stacking was weak or moderate for all induced models. The training set composed by 17 w/b G reached the best correlation and RRSE. However, in some cases where the gain of stacking was high, it was not statistically higher than the best base classifier.

In order to better understand the effects of diversity on the performed experiments, for each dataset where the gain of stacking passed T-test, i.e. where the stacking accuracy was statistically different from the best individual base classifier result, we plotted the values of the diversity measures used in the induced models. Figure 3 shows the relation between df and the gain of stacking. High values of both diversity and gain are plotted in green while low values are plotted in red. These points support the hypothesis that the greater the diversity, the greater the stacking accuracy. Other points plotted in blue go against this hypothesis. Similarly, Figures 4 and 5 present the relationship among KW, ρ and G.

Analyzing these figures we notice that only a few points are good values of diversity. Many of them are located near the center of abscissa axis, as well as the vast majority of low diversity values. The only exception where diversity are strong related to the increase of accuracy occurred with the

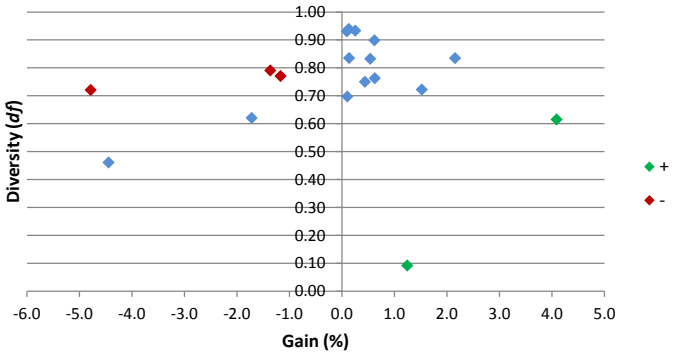


Fig. 3. Relation between double-fault and the gain of stacking.

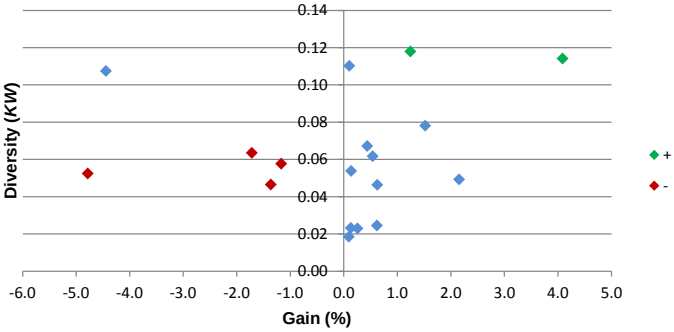


Fig. 4. Relation between Kohavi-Wolpert variance and the gain of stacking.

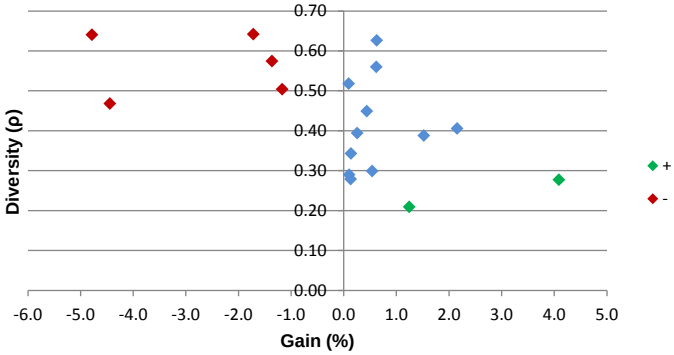


Fig. 5. Relation between correlation coefficient and the gain of stacking.

TABLE IV
EVALUATION OF THE REGRESSION MODELS.

	Datasets	Model	Correlation	RRSE (%)
1	17 b/w G	M5	0.5243	79.67
2	all 54	linear	0.4081	91.58
3	18 T-test	linear	0.3532	99.89

dataset Connect. Bench (S,M vs. R) which experiment reaches $df = 0.62$, $KW = 0.11$, $\rho = 0.28$ and $G = 4.1\%$.

V. CONCLUSION

This paper presented an analysis of the impact of diversity on stacking multiple classifiers. The experiments we have performed show some link between the studied diversity measures and the gain of stacking considering a lot of datasets.

The regression models revealed connections between some measures and the quality of stacking. df, KW and ρ are related to the final classification accuracy, but low values of the correlation coefficients and high values of RRSE imply a weak relationship. So, as suggested by the literature for bagging and majority voting ensembles, predicting the improvement on the best base classifier accuracy using diversity measures is inappropriate.

As future work, we intend to conduct experiments with additional diversity measures and with more synthetic datasets, aiming to better understand the relations between data distribution, decision boundaries, classifiers diversity and the quality of stacking.

REFERENCES

[1] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine Learning*, vol. 40, no. 2, pp. 139–157, 2000.

[2] K. M. Ting and I. H. Witten, "Issues in stacked generalization," *Journal of Artificial Intelligence Research*, vol. 10, pp. 271–289, 1999.

[3] S. Dzeroski and B. Zenko, "Is combining classifiers with stacking better than selecting the best one?" *Machine Learning*, vol. 54, no. 3, pp. 255–273, 2004.

[4] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[5] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.

[6] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine learning*, vol. 51, no. 2, pp. 181–207, 2003.

[7] C. J. Merz, "Using correspondence analysis to combine classifiers," *Machine Learning*, vol. 36, no. 1-2, pp. 33–58, 1999.

[8] S. B. Kotsiantis and P. E. Pintelas, "A hybrid decision support tool-using ensemble of classifiers," in *Proceedings of the International Conference On Enterprise Information Systems*, 2004, pp. 448–453.

[9] S. Ali and A. Majid, "Can-evo-ens," *Journal of Biomedical Informatics*, vol. 54, no. C, pp. 256–269, 2015.

[10] L. Peppoloni, M. Satler, E. Luchetti, C. A. Avizzano, and P. Tripicchio, "Stacked generalization for scene analysis and object recognition," in *Proceedings of the IEEE International Conference on Intelligent Engineering Systems*. IEEE, 2014, pp. 215–220.

[11] N. Larios, J. Lin, M. Zhang, D. Lytle, A. Moldenke, L. Shapiro, and T. Dietterich, "Stacked spatial-pyramid kernel: An object-class recognition method to combine scores from random trees," in *Proceedings of the IEEE Workshop on Applications of Computer Vision*. IEEE, 2011, pp. 329–335.

[12] R. Ebrahimpour, N. Sadeghnejad, A. Amiri, and A. Moshtagh, "Low resolution face recognition using combination of diverse classifiers," in *Proceedings of the International Conference of Soft Computing and Pattern Recognition*. IEEE, 2010, pp. 265–268.

[13] S. R. Ness, A. Theocharis, G. Tzanetakis, and L. G. Martins, "Improving automatic music tag annotation using stacked generalization of probabilistic svm outputs," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2009, pp. 705–708.

[14] C. A. Shipp and L. I. Kuncheva, "Relationships between combination methods and measures of diversity in combining classifiers," *Information Fusion*, vol. 3, no. 2, pp. 135 – 148, 2002.

[15] R. Dymitr and G. Bogdan, "Classifier selection for majority voting," *Information Fusion*, vol. 6, no. 1, pp. 63 – 81, 2005.

[16] B. Sluban and N. Lavrac, "Relating ensemble diversity and performance: A study in class noise detection," *Neurocomputing*, vol. 160, pp. 120 – 131, 2015.

[17] A. T. Muhammad and S. Jim, "Creating diverse nearest-neighbour ensembles using simultaneous metaheuristic feature selection," *Pattern Recognition Letters*, vol. 31, no. 11, pp. 1470–1480, 2010.

[18] M. Makhtar, L. Yang, D. Neagu, and M. Ridley, "Optimisation of classifier ensemble for predictive toxicology applications," in *14th International Conference on Computer Modelling and Simulation*, March 2012, pp. 236–241.

[19] S. Whalen and G. Pandey, "A comparative analysis of ensemble classifiers: Case studies in genomics," in *2013 IEEE 13th International Conference on Data Mining*, Dec 2013, pp. 807–816.

[20] F. A. Faria, J. A. dos Santos, A. Rocha, and R. da S. Torres, "A framework for selection and fusion of pattern classifiers in multimedia recognition," *Pattern Recognition Letters*, vol. 39, pp. 52 – 64, 2014, advances in Pattern Recognition and Computer Vision.

[21] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, USA: Prentice-Hall, Inc., 2007.

[22] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Annual Workshop on Computational Learning Theory*. ACM, 1992, pp. 144–152.

[23] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Proceedings of the Advances in Large Margin Classifiers*. MIT Press, 1999.

[24] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 1995, pp. 338–345.

[25] W. W. Cohen, "Fast effective rule induction," in *Proceedings of the International Conference on Machine Learning*, 1995, pp. 115–123.

[26] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, USA: Morgan Kaufmann Publishers Inc., 1993.

[27] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[28] J. R. Quinlan, "Learning with continuous classes," in *Proceedings of the Australian Joint Conference on Artificial Intelligence*, vol. 92. Singapore: World Scientific, 1992, pp. 343–348.

[29] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognition*, vol. 41, no. 1, pp. 191 – 203, 2008.

[30] C. J. Veenman, M. J. T. Reinders, and E. Backer, "A maximum variance cluster algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1273–1280, Sep 2002.

[31] C. Tomasini, L. Emmendorfer, E. N. Borges, and K. Machado, "A methodology for selecting the most suitable cluster validation internal indices," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2016, pp. 901–903.

[32] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2011.

[33] Student, "The probable error of a mean," *Biometrika*, pp. 1–25, 1908.

Discovering Hidden Interests from Twitter for Multidimensional Analysis

Dongjin Yu, Jingchao Sun, Yiyu Wu, Zhiyong Ni, Youhuizi Li
School of Computer Science and Technology
Hangzhou Dianzi University
Hangzhou,China
E-mail: yudj@hdu.edu.cn, wyygoup@gmail.com

Abstract—With the popularity of social networks, Twitter has become one of the dominant providers of massive quantities of information. Exploring the distributions and correlations from Twitter data helps accurate personalized recommendations. Online Analytical Processing, or OLAP, provides an intuitive form that is suitable for exploring Twitter data. Unfortunately, the traditional OLAP approaches can only deal with structured data, not unstructured textual data like tweets. The key to applying OLAP to Twitter data is to mine and build a dimension hierarchy of tweeter interests. However, the current methods can extract tweeter interests from Twitter data on a single level, but fail to obtain a hierarchy of tweeter interests with different granularities. To address this problem, we propose a LDA-based model, called MS-LDA, which combines tweeters' social relationships and tweets to extract and build the tweeters' interest dimension hierarchy. Such a dimension hierarchy can be further employed to apply OLAP techniques to Twitter data. In addition, we employ Word2vec to obtain the linguistic similarity of words in tweets, to improve its effectiveness. The extensive experiments demonstrate that our method can effectively extract the dimension hierarchy of tweeters' interests for multidimensional analysis.

Keywords: Multidimensional Analysis, Twitters, Tweets, Interests, OLAP, LDA, Tweeters.

I. Introduction

Twitter is an online social networking service that enables tweeters to send and read short messages called "tweets". Technically, Twitter data can be divided into two parts, i.e., structured data such as "id" and "location", and unstructured data which include text messages, short links and so on. In addition, tweeters have some social behaviours, such as Follow (paying attention to others), Mention ("@" others) and Retweet (forwarding others' tweets).

Online Analytical Processing, or OLAP, enables users to analyse multidimensional data interactively from multiple perspectives using operations such as roll-up, drill-down, slicing and dicing [1]. With the explosive growth of Twitter, it has become increasingly necessary to introduce the technology of OLAP to analyse complex Twitter data in an interactive manner. Unfortunately, although OLAP technology provides an intuitive inquiry form that is consistent with human custom, it can only handle structured data, and fails to deal with scenarios related to unstructured text data such as tweets. Therefore, the key to applying OLAP to Twitter data is the question of how to identify dimensions from Twitter data. Among these, the interest dimension attracts the most attention from researchers.

Although some progress has been made in the area of online analysis of Twitter data, there are still some problems to be solved. Many state-of-the-art approaches can only identify a single dimension from tweets instead of the whole dimension hierarchy. Although some of the approaches are able to identify dimension hierarchies, this usually depends on the efforts of domain experts. The main issue this paper tries to resolve can be described as follows: *how to extract the relevant information from the unstructured Twitter data to construct dimension hierarchies of tweeters' interests to achieve effective online multidimensional analysis.*

In this paper, we propose a multi-layered semantic LDA (MS-LDA) model to achieve multidimensional analysis of Twitter data. Firstly, we acquire tweets and the social relationship among tweeters through the REST APIs provided by Twitter. Secondly, we utilize data pre-processing techniques, such as ignoring stop words and part of speech analysis, to remove the irrelevant words in tweets. Finally, we extract the dimension hierarchy of tweeters' interests based on the probability distribution of the various interests and sub-interests revealed in the tweets. To summarize, the multi-layered semantic LDA (MS-LDA) proposed in this paper can not only extract tweeters' interests, but also dig out more fine-grained sub-interests. The interests and their sub-interests thus constitute the dimensional hierarchy of interests and the final data cube together with the

DOI reference number: 10.18293/SEKE2017-131

other structured dimensions.

The main contributions of this paper are as follows. 1) We present a new model called MS-LDA for mining the dimension hierarchy of a tweeter's interests from unstructured massive Twitter data, which considers not only tweets but also the social relations among tweeters. 2) We take advantage of Word2vec, a two-layer neural network model, to obtain the linguistic similarity of words to improve the effectiveness of identifying the interest dimensions. 3) We perform the multidimensional analysis of Twitter data which demonstrates the effectiveness of MS-LDA.

The rest of the paper is organized as follows. After Section 2 introduces related work, Section 3 describes the approach in detail, focusing on the interest mining and the construction of interest dimension hierarchies. In Section 4, we show the overall effect on real Twitter data, and compare our approach with others. Finally, Section 5 concludes the paper and outlines future work.

II. Related Work

During the past decade, with the increased amount of text data, such as web pages, tweets and web blogs, the question of how to apply the traditional OLAP technique to the unstructured text for business intelligence has attracted much attention from many researchers.

Numerous studies have been conducted to explore the social networks such as Twitter. Siswanto et al. [2] use the supervised learning-based classification to determine the user's interests based on the bio and a collection of tweets. Lim et al. [3] present a framework for classifying the relative interests of Twitter users using information from Wikipedia. Pu et al. [4] present Wiki-LDA to mine user's interests in Twitter. Xu et al. [5] propose a framework to discover users' interests by introducing a modified author-topic model called the twitter-user model. Zhao et al. [6] introduce a Twitter-LDA model which is specifically used to extract topics from short tweet data.

Meanwhile, many researchers concentrate on extending the traditional multidimensional data model to support unstructured text such as Twitter data. For example, Maha et al. [7], propose a generic multidimensional model dedicated to the OLAP of tweets whereas N. Rehman et al. [8] develop a system for warehousing streams from Twitter. These authors also attempt to extend the established OLAP technology to enable multidimensional analysis of social media data by integrating text and opinion-mining methods into the data warehousing system [9].

In contrast to previous work, the Multi-layered Semantic LDA (MS-LDA) model proposed in this paper can not only extract tweeters' interests but also dig out more fine-grained sub-interests. The interests and the corresponding sub-interests thus constitute the dimensional hierarchy and the final data cube for multi-dimensional analysis.

III. Mining Interests

This section discusses how to construct MS-LDA to mine tweeters' interests from Twitter data, and how to construct interest dimension hierarchies based on the mined interests.

A. Overview of MS-LDA

LDA is a document topic generation model, which denotes that each word in the document is obtained through a particular process, that is, choosing a topic with a certain probability, and then choosing a word from this topic with a certain probability [10]. Unfortunately, due to the similarities existing between some topics, LDA cannot distinguish between some fine-grained ones. In addition, LDA does not consider the semantic features of words. In a similar way to LDA, the words and documents in MS-LDA are associated with a potential topic. However, using MS-LDA it is possible to identify multi-level topics hidden in large-scale data such as Twitter, which LDA fails to do. Moreover, MS-LDA allows the semantic similarity between words and interests to affect the process of model generation, in order to improve the recognition results of sub-interests. Here, we employ the term "interest" instead of the term "topic" to better fit the current application domain.

Tweets are different from the general texts. In addition to text information, the tweets also contain complex social relationships, such as Following, Mentioning and Retweeting [11]. To capture the above social relationships, we build a social list for each tweeter, including the original tweeters of forwarded tweets, the mentioned tweeters and the followed tweeters. We then add this social list as an influence factor into the MS-LDA model.

In summary, MS-LDA integrates tweets and the social relationships among tweeters, and is therefore suitable for multi-level interest mining. The Bayesian network graph of MS-LDA is shown in Figure 1. When MS-LDA generates a tweet, it first chooses an interest from the multinomial distribution θ_m which is generated by Dirichlet distribution with parameter α. Next, it chooses a sub-interest $z^{'}$ according to the degree of social impact and semantic impact. Finally, it chooses a word w from the multinomial distribution $\varphi_{k'}^{'}$ which is generated by Dirichlet distribution with parameter $\beta^{'}$ and $Y^{'}$. For simplicity, Table 1 summarizes the notations used throughout this paper.

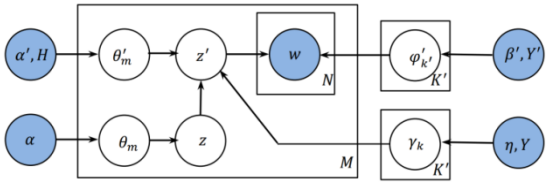


Figure 1: The Bayesian network of MS-LDA

Table 1: Notations used throughout the paper

Notation	Definition
K	Number of interests
M	Number of tweeters
V	Number of words
$K^{'}$	Number of sub-interests
θ_m	Interest distribution over the tweeter m based on tweets
$\theta_m^{'}$	Sub-interest distribution over the tweeter m based on tweets
$\varphi_{k'}^{'}$	Word distribution over sub-interest $k^{'}$
γ_k	Sub-interest distribution over interest k
η	Hyper parameter for $\gamma_k, 1 \leq k \leq K$
α	Hyper parameter for $\theta_m, 1 \leq m \leq M$
$\alpha^{'}$	Hyper parameter for $\theta_m^{'}, 1 \leq m \leq M$
$\beta^{'}$	Hyper parameters for $\varphi_{k'}^{'}, 1 \leq k^{'} \leq K^{'}$
w	Word in tweets
$z_{m,v}$ $(z_{-m,v})$	Interest of word v in tweets of tweeter m (indicator before sampling)
$z_{m,v}^{'}$ $(z_{-m,v}^{'})$	Sub-interest of word v in tweets of tweeter m (indicator before sampling)
$n_{m,k'}$ $(n_{m,\cdot}^{'})$	Co-occurrence of tweeter m and sub-interest $k^{'}$ (or all sub-interests)
$n_{k,v}$ $(n_{k,\cdot})$	Co-occurrence of interest k and word v (or all words)
$n_{k,k'}$ $(n_{k,\cdot}^{'})$	Co-occurrence of interest k and sub-interest $k^{'}$ (or all sub-interests)
$n_{k',v}$ $(n_{k',\cdot})$	Co-occurrence of sub-interest $k^{'}$ and word v (or all words)

In other words, for a given tweet, its joint probability distribution of all the words and their interests and sub-interests can be calculated as follows:

$$P\left(z, z^{'}, w \middle| \alpha, \alpha^{'}, \beta^{'}, \eta, H, Y, Y^{'}\right) = P(z|\alpha)P(z^{'}|z, \alpha^{'}, \eta, H, Y)P(w|z^{'}, \beta^{'}, Y^{'}) \tag{1}$$

B. Derivation of MS-LDA

The MS-LDA model can be derived using the Gibbs sampling method. The main steps are as follows:

Firstly, the interest probability distribution $P(z|\alpha)$, the sub-interest probability distribution $P\left(z^{'} \middle| z, \alpha^{'}, \eta, H, Y\right)$ and the word probability distribution $P(w|z^{'}, \beta^{'}, Y^{'})$ can be calculated according to the Euler formula, as follows:

$$P(z|\alpha) = \left(\frac{(\Gamma(K\alpha)}{\prod_k \Gamma(\alpha)}\right)^T \prod_{m=1}^{M} \frac{\prod_k \Gamma(n_{m,k}+\alpha)}{\Gamma(n_{m,\cdot}+K\alpha)} \tag{2}$$

$$P\left(z^{'} \middle| z, \alpha^{'}, \eta, H, Y\right) = \left(\frac{(\Gamma(K^{'}\alpha^{'})}{\prod_{k^{'}} \Gamma(\alpha^{'})}\right)^T \prod_{m=1}^{M} \frac{\prod_{k^{'}} \Gamma\left(H_{m,k^{'}}\left(n_{m,k^{'}}+\alpha^{'}\right)\right)}{\Gamma\left(H_{m,\cdot}\left(n_{m,\cdot}^{'}+K^{'}\alpha^{'}\right)\right)} \times \left(\frac{(\Gamma(K^{'}\eta)}{\prod_{k^{'}} \Gamma(\eta)}\right)^T \prod_{k=1}^{K} \frac{\prod_{k^{'}} \Gamma\left(Y_{k,k^{'}}(n_{k,k^{'}}+\eta)\right)}{\Gamma\left(Y_{k,\cdot}(n_{k,\cdot}^{'}+K\eta)\right)} \tag{3}$$

$$P\left(w \middle| z^{'}, \beta^{'}, Y^{'}\right) = \left(\frac{(\Gamma(V\beta^{'})}{\prod_v \Gamma(\beta^{'})}\right)^T \prod_{k^{'}=1}^{K^{'}} \frac{\prod_v \Gamma\left(Y_{k^{'},v}^{'}(n_{k^{'},v}+\beta^{'})\right)}{\Gamma\left(Y_{k^{'},\cdot}^{'}(n_{k^{'},\cdot}+V\beta^{'})\right)} \tag{4}$$

The interpretation of the $H_{m,k'}Y_{k',v}^{'}$ and $Y_{k,k'}$ in the above equations is as follows.

The social behaviour among tweeters reflects their interest preferences to a certain extent. $H_{m,k'}$, or the social impact, denotes the degree to which the social relationship affects the tweeter m being interested in the sub-interest $k^{'}$. Let $P_{m,k'}$ be the probability of tweeter *m* being interested in the sub-interest $k^{'}$, which can be obtained by Eq. (3), and let $S_m = \{u_1, u_2, u_3, \cdots, u_{N_m}\}$ be the social list of tweeter m in which u_j represents the *j-th* tweeter in the social list S_m and N_m represents the total number of tweeters in the social list S_m. The *social impact* can thus be calculated according to Eq. (5), where $P_{-u_j,k'}$ represents the probability that the tweeter u_j is interested in the $k^{'}$ sub-interest in the previous iteration.

$$H_{m,k'} = \frac{\sum_{j=1}^{N_m} P_{-u_j,k'}}{N_m} \tag{5}$$

We consider that the higher the semantic similarity between words and the sub-interest, the greater the probability that the word belongs to the sub-interest. Let $Y_{k',v}^{'}$ be the degree of *word/sub-interest semantic impact* of the word v belonging to the sub-interest $k^{'}$. For the computation of word/sub-interest semantic impact, we pick up the most frequent $n^{'}$ words which belong to the sub-interest $k^{'}$ to compose a collection $Q_{k'}^{'} = \{q_{k',1}^{'}, q_{k',2}^{'}, \ldots, q_{k',n'}^{'}\}$. Let $R_{k'}^{'} = \{r_{k',1}^{'}, r_{k',2}^{'}, \ldots, r_{k',n'}^{'}\}$ be the number-of-the-words collection in which each item gives the number of occurrences of the corresponding word, and $Sim(v, q_{k',j}^{'})$ be the semantic similarity between the word v and $q_{k',j}^{'}$. The *word/sub-interest semantic impact* can thus be obtained from Eq. (6).

$$Y_{k',v}^{'} = \frac{\sum_{j=1}^{n^{'}} \left(r_{k',j}^{'} \bullet Sim\left(v, q_{k',j}^{'}\right)\right)}{\sum_{j=1}^{n^{'}} r_{k',j}^{'}} \tag{6}$$

For example, when calculating the degree of semantic impact of the word "Yao_Ming" (short to v) belonging to the sub-interest P, we first pick up the most frequent top three words which belong to the sub-interest P, i.e., $Q_P^{'} = \{\text{Basketball}, Kobe\_Bryant, NBA\}$. Suppose $R_P^{'} = \{30, 20, 10\}$ is the number of occurrences of the corresponding words. Thus, the degree of *word/sub-interest semantic impact* of the word "Yao_Ming" belonging to the sub-interest P can be calculated as:

$$Y_{P,v}^{'} = \frac{(30 * 0.35 + 20 * 0.63 + 10 * 0.42)}{(30 + 20 + 10)} = 0.455.$$

To calculate $Sim(v, q_{k',j}^{'})$, we utilize Word2vec, an efficient tool proposed by Google to characterize a word as a vector [12]. According to Word2vec, the similarity in the vector space can be used to represent the similarity of text semantic. For example, we can obtain the vectors of "Basketball", "Football", "Yao_Ming" and "Kobe_Bryant". The similarity of "Yao_Ming" and "Basketball" calculated by Word2vec is higher than that of "Yao_Ming" and "Football", which is consistent with the real case.

For the words w_1 and w_2, we firstly obtain their vectors V_1 and V_2 respectively by Word2vec. Then, their semantic similarity can be calculated as follows:

$$Sim\,(w_1, w_2) = \frac{\sum_{i=1}^{x}(V_{1,i} \times V_{2,i})}{\sqrt{\sum_{i=1}^{x} V_{1,i}^2} \times \sqrt{\sum_{i=1}^{x} V_{1,i}^2}} \tag{7}$$

In a similar way to the *word/sub-interest semantic impact*, the higher the semantic similarity between sub-interest and interest, the greater the probability that the sub-interest belongs to the interest. Let $Y_{k,k'}$ be the degree of *sub-interest/interest semantic impact* of the sub-interest $k^{'}$ belonging to the interest k. We select the most frequent n words which belong to the interest k to compose a collection $Q_k = \{q_{k,1}, q_{k,2}, \ldots, q_{k,n}\}$. Suppose $R_k = \{r_{k,1}, r_{k,2}, \ldots, r_{k,n}\}$ is a collection such that each item is the number of occurrences of the corresponding word. The *sub-interest/interest semantic impact* can thus be calculated as:

$$Y_{k,k'} = \frac{\sum_{i=1}^{n} r_{k,i} \frac{\sum_{j=1}^{n^{'}} r_{k',j}^{'} Sim(q_{k,i},\ q_{k',j}^{'})}{\sum_{j=1}^{n^{'}} r_{k',j}^{'}}}{\sum_{i=1}^{n} r_{k,i}} \tag{8}$$

We then sample all interests and sub-interests until we achieve the stable sampling results according to the following two posterior distributions:

$$\begin{aligned} P\left(z_{m,v}^{'} \middle| w, \alpha^{'}, \beta^{'}, z_{-}^{'}, H, Y^{'}\right) &= \frac{P\left(z^{'}, w \middle| \alpha^{'}, \beta^{'}, H, Y^{'}\right)}{P\left(z_{-}^{'}, w \middle| \alpha^{'}, \beta^{'}, H, Y^{'}\right)} \\ &\propto \frac{Y_{k',v}^{'}\left(n_{k',v}^{'} + \beta^{'}\right)}{Y_{k',\cdot}^{'}\left(n_{k',\cdot}^{'} + V\beta^{'}\right)} \times \frac{H_{m,k'}\left(n_{m,k'}^{'} + \alpha^{'}\right)}{H_{m,\cdot}\left(n_{m,\cdot}^{'} + K^{'}\alpha^{'}\right)} \end{aligned} \tag{9}$$

$$\begin{aligned} P\left(z_{m,v} \middle| z^{'}, z_{-}, \alpha, Y, \eta\right) &= \frac{P\left(z, z^{'} \middle| \alpha, Y, \eta\right)}{P\left(z^{'}, z_{-} \middle| \alpha, Y, \eta\right)} \\ &\propto \frac{Y_{k,k'}\left(n_{k,k'}^{'} + \eta\right)}{Y_{k,\cdot}\left(n_{k,\cdot}^{'} + K\eta\right)} \times \frac{n_{m,k} + \alpha}{n_{m,\cdot} + K\alpha} \end{aligned} \tag{10}$$

Finally, we obtain:

$$\theta_m = \frac{n_{m,k} + \alpha}{n_{m,\cdot} + K\alpha} \tag{11}$$

$$\theta_m^{'} = \frac{H_{m,k'}\left(n_{m,k'}^{'} + \alpha^{'}\right)}{H_{m,\cdot}\left(n_{m,\cdot}^{'} + K^{'}\alpha^{'}\right)} \tag{12}$$

$$\varphi_{k'}^{'} = \frac{Y_{k',v}^{'}\left(n_{k',v}^{'} + \beta^{'}\right)}{Y_{k',\cdot}^{'}\left(n_{k',\cdot}^{'} + V\beta^{'}\right)} \tag{13}$$

$$\gamma_k = \frac{Y_{k,k'}\left(n_{k,k'}^{'} + \eta\right)}{Y_{k,\cdot}\left(n_{k,\cdot}^{'} + K\eta\right)} \tag{14}$$

C. Constructing Interest Dimension Hierarchies

Following the above steps, we can obtain the tweeter-interest probability distribution θ_m, tweeter-sub-interest probability distribution $\theta_m^{'}$, sub-interest-word probability distribution $\varphi_{k'}^{'}$, and interest-sub-interest probability distribution γ_k. Now, we exploit these matrices to construct the interest dimension hierarchy to support OLAP.

First, we determine the relationship between sub-interests and interests according to the distribution of the interest-sub-interest probability distribution γ_k. Secondly, a word can belong to multiple sub-interests according to the distribution of sub-interest-word $\varphi_{k'}^{'}$. We choose the top 20 maximum probability words that belong to the sub-interest to be the word set of this sub-interest. Finally, we name the sub-interest according to its corresponding word set, and the interest according to the sub-interest set that belongs to it.

IV. Experiment

A. Experimental Setup

To evaluate the effectiveness of our approach, we conducted extensive experiments on real Twitter data collected by Twitter Rest API. Firstly, we selected 15 Twitter users from the Twitter homepage as the seeds and acquired all their 7,373 followers. Next, we retrieved the tweeters' profiles (nicknames, locations, etc.), tweets (text, favourite count, retweet count, etc.), and their social relationships (follower list, fan list, etc.). Finally, we removed the tweets with fewer than six words and the duplicate tweets. In this way, we obtained 10,160,317 tweets from 6,907 tweeters.

Table 2 shows the statistics of the dataset. The number of tweets that contains "@" amounts to 62% of the total and the average number of twitter's followers is 1565, indicating the rich social relationships.

Table 2: Dataset Statistics

	#Twetters	#Tweets	#Tweets with @	#Topics	#Words	#Distinct words	#Follower
Total	6,907	10,160,317	6,301,995	571,068	175,040,499	541,430	10,810,638
Average of twitters	—	1,471	912	83	25,342	78	1,565
Average of tweets	—	—	62%	—	17.2	—	—

The vector model of Word2vec was trained using the Google News data set, which consists of 100 billion words and covers three million words and phrases, each of which

has 300 dimensions. The experiments ran on a server with six cores of E5-2620 2.00GHz, 64GB memory, and the Windows 7 operating system.

B. Evaluation of Sub-interest Identification

We randomly selected 110 tweeters from the total of 6,907. Five postgraduate students manually labelled the sub-interests of the 110 tweeters independently. If they failed to reach the same sub-interests for a given tweeter, they simply labelled the tweeter with the majority vote. The performance of MS-LDA was evaluated using three widely-adopted metrics, namely precision, recall and F-measure.

$$\text{precision}_i = 100\% \times \frac{|tweeters\ correctly\ identified\ to\ i{-}th\ interest|}{|tweeters\ identified\ to\ i{-}th\ interest|} \tag{15}$$

$$\text{recall}_i = 100\% \times \frac{|tweeters\ correctly\ identified\ to\ i{-}th\ interest|}{|tweeters\ actually\ belonging\ to\ i{-}th\ interest|} \tag{16}$$

$$\text{f-measure}_i = 100\% \times \frac{2 \times \text{precision}_i \times \text{recall}_i}{\text{precision}_i + \text{recall}_i} \tag{17}$$

$$\text{f-measure} = \frac{1}{n}\sum_{i=1}^{n} \text{f-measure}_i \tag{18}$$

A tweeter often has more than one interest. We consider that a tweeter has a particular interest if his probability of having this interest is greater than a predefined threshold value. Here, we set this threshold as 0.1, or T=0.1. To evaluate its effectiveness, we compared our model with LDA and Twitter-LDA [6]. We manually labelled the sub-interests based on the results obtained by LDA and Twitter-LDA since neither LDA nor Twitter-LDA provides multi-level interests. Because a poor result would lead to a poor labelling of sub-interests and vice versa, this comparison can partially demonstrate the performance of our approach. Table 3 shows the average precision (P), recall (R) and F-measure (F) of MS-LDA compared with those of LDA and Twitter-LDA. As it indicates, our model achieves the best values.

Table 3: Comparisons of MS-LDA, Twitter-LDA and LDA

	LDA			Twitter-LDA			MS-LDA		
	P%	**R%**	**F%**	**P%**	**R%**	**F%**	**P%**	**R%**	**F%**
Average	68.69	73.82	69.89	75.09	81.14	76.18	82.24	87.11	82.96

Unlike Twitter-LDA, MS-LDA considers the social relationships among tweeters and the word semantics of tweets. In order to investigate the effect of these two factors, we also conducted experiments without these factors. As Figure 2 shows, the F-measures of standard MS-LDA are better than those of MS-LDA which does not consider social relationships or word semantics under all circumstances. On the other hand, the F-measure of MS-LDA without considering tweeters social relationships achieves 79.54%, which is lower than that of the standard MS-LDA but higher than Twitter-LDA. Meanwhile, the total F-measure of MS-LDA without considering tweets' word semantics achieves 72.48%, which is far lower than that of the standard MS-LDA and Twitter-LDA, but higher than the traditional LDA.

Table 4 shows the statistical significance based on p value of MS-LDA compared with those of LDA and Twitter-LDA. In the case of the significance value being set to 0.1, i.e., $\alpha = 0.1$, our method outperforms the other two with regard to precision, recall and F-measure.

Table 4: Statistical Significance Between MS-LDA and LDA

	P	**R**	**F**
P Value: MS-LDA vs LDA	0.0954	0.0934	0.0901
P Value: MS-LDA vs TWITTER-LDA	0.0951	0.0934	0.0951

C. Overall Effect

Based on the above data model, we can conduct various operations such as roll-up, drill-down and dicing on Twitter data. For example, Figure 3 shows the result when drilling down from "Sports", which reveals the distribution of three different sub-interests of "Sports" in different areas. The data can be aggregated by rolling the area level up to the country level based on the "Location" dimension.

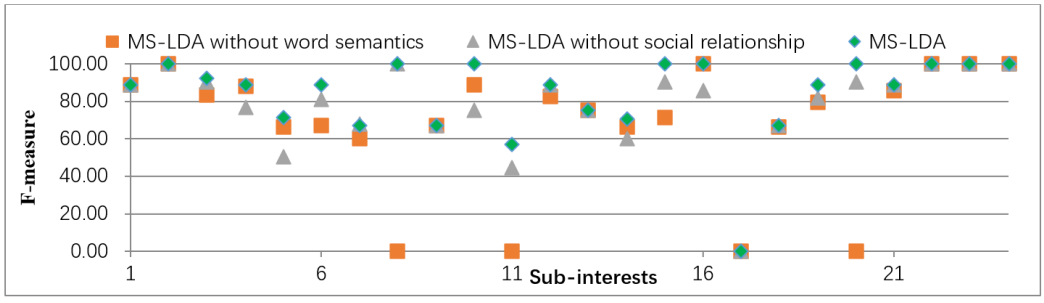


Figure 2: The effectiveness of considering social relationship and word semantics

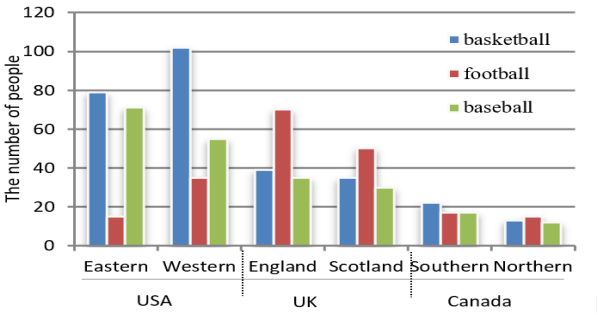


Figure 3: Distribution of sports interests in different locations

Generally speaking, the interests that tweets reveal are always changing, indicating a drift with time. Because all the tweeters' tweets are grouped according to different granularities with respect to time periods, the trained MS-LDA model can be employed to discover the drift of interest distribution of the tweeters in different periods. As Figure 4 demonstrates, in January and February tweeters pay more attention to basketball, but in April and May their interests are transferred to football. On the other hand, the numbers of tweeters who are concerned about movies remains almost stable from January to February in 2016.

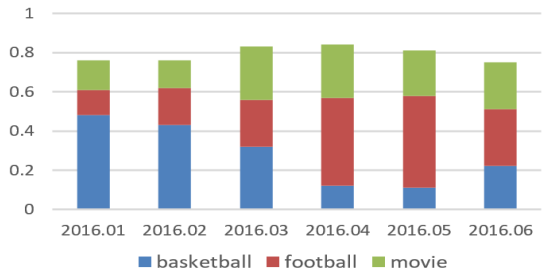


Figure 4: Changes in tweeter interests

V. Conclusion and Future Work

In this paper, we propose an improved topic model, i.e., MS-LDA, which is used to extract the dimension hierarchies of tweeters' interests, normally hidden in the large amount of unstructured Twitter data. We conducted extensive experiments on a large real data set collected by Twitter API to evaluate the effectiveness of MS-LDA. The results show that MS-LDA has a better interest recognition effect than other models.

The Word2vec model employed in this paper is trained using news provided by Google. However, the presentation of news in general is somewhat rigorous, while tweets are more colloquial. In the future, we will consider the use of Twitter data to train the Word2vec model to improve the effectiveness of MS-LDA. In addition, we plan to parallel MS-LDA to improve the running speed.

ACKNOWLEDGMENT

The work is supported by National Natural Science Foundation of China (No. 61100043), the Key Science and Technology Project of Zhejiang Province (No. 2017C01010, No. 2016F50014 and No. 2015C01040). The authors would also like to thank anonymous reviewers who made valuable suggestions to improve the quality of the paper.

References

[1] S. Mansmann, N. U. Rehman, A. Weiler, and M. H. Scholl, "Discovering olap dimensions in semi-structured data," *Information Systems*, vol. 44, pp. 120–133, 2014.

[2] E. Siswanto, M. L. Khodra, and L. J. E. Dewi, "Prediction of interest for dynamic profile of twitter user," in *Advanced Informatics: Concept, Theory and Application (ICAICTA), 2014 International Conference of*. IEEE, 2014, pp. 266–271.

[3] K. H. Lim and A. Datta, "Interest classification of twitter users using wikipedia," in *Proceedings of the 9th International Symposium on Open Collaboration*. ACM, 2013, p. 22.

[4] X. Pu, M. A. Chatti, U. Schroeder *et al.*, "Wiki-lda: A mixed-method approach for effective interest mining on twitter data," in *Proceedings of CSEDU 2016*, no. EPFL-CONF-217479, 2016.

[5] Z. Xu, L. Ru, L. Xiang, and Q. Yang, "Discovering user interest on twitter with a modified author-topic model," in *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01*. IEEE Computer Society, 2011, pp. 422–429.

[6] 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.

[7] M. B. Kraiem, J. Feki, K. Khrouf, F. Ravat, and O. Teste, "Olap of the tweets: from modeling toward exploitation," in *Proceedings of the 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS)*. IEEE, 2014, pp. 1–10.

[8] N. U. Rehman, S. Mansmann, A. Weiler, and M. H. Scholl, "Building a data warehouse for twitter stream exploration," in *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*. IEEE Computer Society, 2012, pp. 1341–1348.

[9] N. U. Rehman, A. Weiler, and M. H. Scholl, "Olaping social media: the case of twitter," in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ACM, 2013, pp. 1139–1146.

[10] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[11] M. K. Olorunnimbe and H. L. Viktor, "Tweets as a vote: Exploring political sentiments on twitter for opinion mining," in *Proceedings of the International Symposium on Methodologies for Intelligent Systems*. Springer, 2015, pp. 180–185.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Computer Science*, 2013.

Software Defect Prediction Using Dictionary Learning

Hongyan Wan[1,2], Guoqing Wu[1,2], Ming Cheng[1,2], Qing Huang[1,2], Rui Wang[1,2] and Mengting Yuan[1,2*]
1. State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China
2. School of Computer, Wuhan University, Wuhan 430072, China
E-mail: { why0511, wgq, chengming, qh,wangrui1989,ymt }@whu.edu.cn

***Abstract*—With the popularization of software version control system and defect tracking tools, large amounts of software development data is recorded. How to effectively use these data to improve the quality of software development, has become a hot topic in recent years. Software defect prediction technology can take full advantage of the historical data to build predictive models and automatically detect defective modules for efficient software test to improve the quality of a software system. But the class-imbalanced data makes the prediction model classifying a modules as a defective-free one easily, while the misclassification of defective modules generally incurs much higher cost risk than the misclassification of defective-free ones. To resolve this problem, we propose a cost-sensitive software defect prediction method using dictionary learning. It iteratively optimizes the classifier parameters and the dictionary atoms, to ensure that the extracted features (sparse representation) are optimal for the trained classifier; Moreover, we take the different misclassification costs into account, increasing the punishment on misclassification defective modules in the procedure of dictionary learning, making classification inclining to classify a module as a defective one. Experimental results on the 10 class-imbalanced data sets of NASA show that our method is more effective than other methods.**

Keywords- Software defect prediction; Dictionary learning; Cost-sensitive; Bilevel optimization; Sparse coding

I. INTRODUCTION

In recent years, soft version control system and defect tracking system have widely used in software project, so that each code submission, each defect report comments are fully recorded. How to make use of these data to measure the efficiency, calculate the cost, and predict the quality of the products becoming a hot topic in the software development of big data background [1]. The software defect prediction technology can take full advantage of the software historical data to build the prediction model, which can accurately predicts whether the software module contains the defect and allocate the test resource to analyze the defective module in the early stage of the system development, thus reducing the software development and maintenance cost [2].

The traditional software defect prediction mainly uses the classification technology of machine learning [3][4], generally divided into two stages: the first stage is the feature extraction, analysis the attribute feature of input software model (such as Halstead, McCabe and others), generate feature vector. The second stage is the defect identification, using the previously trained classifier model to discriminate whether the input software module is defective. Commonly used method include Support Vector Machine (SVM) [5], Naive Bayesian (NB) [6], Neural Network [7] and so on. The performance of the traditional method is seriously limited by the software features, and the feature selection is the difficulty of software defect prediction, lack of common feature representation and feature selection algorithms. Through in-depth research on the characteristics of defect prediction tasks and defect data, we can find that sparseness is a general nature of defect prediction: 1) in general, the defect module is distributed sparsely in the software data set, that is, the number of defective modules is far less than defective-free modules in the software system, which makes the defect data has the class-imbalance characteristics[8][9]; 2) the software module to be tested is usually similar to the historical data of the project and belongs to the same, so it can be sparsely represented by a specific similar historical software module.

With the development of theory and application in Sparse Representation based Classification (SRC), it has been widely applied in pattern recognition, image classification, defect prediction, and so on[10][11][12]. Compared with the traditional classification algorithms, SRC has better discrimination and robustness, it consists of three parts: one is over-complete dictionary; the second is the linear representation under a specific sparse constraints based on the over-complete dictionary; the third is the classification according to the over-complete dictionary and the sparse coefficient of the input data. Wright et al. [13] proposed a SRC-based face recognition method that achieved good results. Jing et al. [14] are the first to apply the dictionary learning technology to the field of software defect prediction, and achieved good results by using the over-complete dictionary prediction defect software module. However, the classification loss function of the method was defined as data sample reconstruction error, making the classification performance mainly dependent on the quality of the training data.

Based on the sparseness principle of software defects, this paper proposes a software defect prediction model based on cost-sensitive dictionary learning (CSDL) under the background of large data, and designs a projection first-order stochastic gradient descent algorithm based on sparseness. The algorithm solves the objective function and verifies the validity of the above model and algorithm on the 10 class-imbalanced data sets provided by NASA. The method is divided into two stages: 1) training the over-complete dictionary, extract the features (sparse coding); 2)

* Corresponding author.
DOI reference number: 10.18293/SEKE2017-188

training classifier according to the extracted features. Training dictionary to solve the sparse coding or training classifier parameters alone cannot get the ideal classification effect. In this paper, we propose a method to iteratively optimize the parameters of the classifier and the dictionary atom to ensure the feature extraction (sparse representation) under the optimal classification performance. At the same time, we take full account of the impact of different misclassification in the process of learning the dictionary, improve the cost of misclassifies a defective module as defective-free one, so that the resulting dictionary is more inclined to classification defect module. Because sparseness is a kind of non-related and versatile features, the model and algorithm proposed in this paper are superior to other traditional methods.

II. RALATED WORK

The purpose of software defect prediction is to automatically identify software modules that contain defects, which are critical to ensuring software quality.

Menzies et al. [15] argue that software engineering data sets are class-imbalanced, using data sampling and lifting algorithms to improve the performance of the model. However, the sampling strategy will change the distribution of the source data, affecting the effect of the prediction. Sun et al. [16] proposed a coding-based integrated learning approach that transforms the class-imbalanced defect data into multi-class balanced data to avoid the class-imbalance problem by specific coding strategies. Jiang et al. [17] pointed out that the performance of the defect prediction model is affected by different misclassification cost. In this paper, on the one hand, cost-sensitive learning method is used to produce the smallest misclassification cost, which makes it easy to classify the defective module with higher misclassification cost, and reduce the class-imbalance problem. On the other hand, considering the different misclassification cost, so that the prediction model is more inclined to classify the defect module, thus improve the prediction performance of the model.

In the sparse representation based classification method, it is necessary to set the dictionary in advance. Wright et al. [13] used all classes of training samples as a dictionary coding test sample, and classify the test samples through minimizing the reconstruction errors. However, the dictionary which they proposed cannot represent test data efficiently due to the presence of noise in the original data set. In addition, the number of dictionary atoms is too large to increase the complexity of coding. Mariral et al. [18] proposed a discriminant dictionary learning method by training the classifier coding coefficients and validating them in the digital recognition and texture classification. As the classification loss function in the SRC is usually defined as a reconstruction error of data samples. Therefore, the performance of the classifier mainly depends on the quality of training data and cannot achieve good classification effect. Jiang et al. [19] integrated the identification of sparse coding error and classification error into a target function to improve the representation and discriminant of the dictionary, joint the learning dictionary with a linear classifier. Although the above methods can achieve better classification accuracy in the corresponding fields, the imbalanced defect data in the field of software defect prediction makes the learning dictionary tendency classification module as defect-free, and the cost of the misclassified defect module is far more than the defect-free module. Therefore, full consideration of this information can be more effective in improving the performance of defect prediction.

In this paper, we optimize the classifier parameters and dictionary atoms iteratively in the process of dictionary learning, to ensure that the feature (sparse representation) is extracted under the optimal classification performance.

III. COST-SENSITIVE DICTIONARY LEARNING METHOD FOR BIG DATA

This section describes the defect prediction process of CSLD method in two stages, and the concrete process is shown in Figure 1. The first stage is to optimize dictionary and the classifier parameters iteratively; the second stage is the classification. In software defect prediction, a test module is usually similar to a partial history module and belongs to the same class (defect or defect-free class), that is, the test module can be compressed as a linear combination of a small number of the same class of modules in historical training data, this representation is usually sparseness. However, classification loss function in the SRC is usually defined as the reconstruction error of the data samples, it is heavily rely on the data quality, and cannot achieve good classification effect. Therefore, this paper presents a dictionary learning method, which optimize the dictionary atoms and classifier parameters iteratively, and applies it to the task of software defect prediction. In addition, there are two types of misclassification errors in the software defect prediction. The type I misclassifies a defective-free module as defective module, while the type II misclassified a defective module as defective-free module, the cost of them is different. In the process of dictionary learning, we fully consider the misclassification costs of type I and II errors to reduce the minimum misclassification cost and improve the prediction performance.

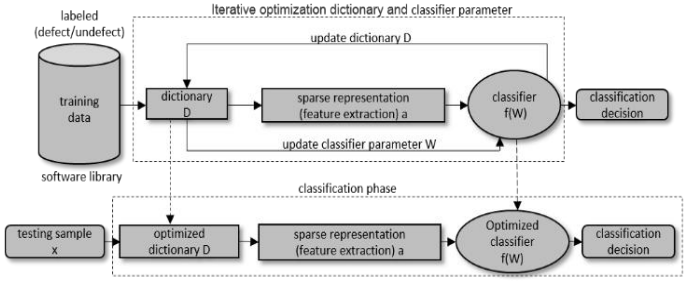


Figure 1. The predict process of CSDL model

The main process of CSLD method is shown in Figure 1. First, using KSVD [20] technology to initialize over-complete dictionary D. In the iterative optimization stage, the sparse coding (feature extraction) a of training samples x in over-complete dictionary D is solved and used to optimize the parameters of the classifier; at the same time, in order to ensure that the sparse coding is under the premise of the optimal classification performance, we apply the algorithm 1 which is proposed in this paper to update

the over-complete dictionary iteratively. In the test classification phase, the test sample solves the sparse coding in the optimized dictionary and inputs it into the optimization classifier. We use logic loss function as the objective function of the classifier.

Sparse coding depends on the over-complete dictionary, its representation is $D=(d_1,d_2,\cdots,d_p),d_i\in\mathbb{R}^m(m\ll p)$, where d_i represents a base vector in the dictionary, referred to as atom. Each sample x can be combined linearly by dictionary atoms, $a\in\mathbb{R}^p$ is the coefficient of sparse representation of x in the dictionary, it is solved by Elastic Net [21].

A. Feature Extraction based on Sparse Coding

Assuming that the software module sample space is $\mathcal{X}$, the label space is $\mathcal{Y}$, the training sample set is $X=\{x_1,x_2,\ldots,x_n\}\in\mathbb{R}^{m\times n},X\in\mathcal{X}$, where n is the number of sample modules, m is the number of attributes of the sample. The module label $y_i\in Y$ indicates the label of the software module x_i. In the software defect prediction model $Y\in\mathcal{Y}=\{+1,-1\}$, where "+1" denotes a defective module, "-1" denotes a defective-free module. Given training sample set $L=\{(x_1,y_1),\cdots,(x_i,y_i)\}(i=1,\cdots,n)$, $D=(d_1,d_2,\cdots,d_p)\in\mathbb{R}^{m\times p}$ is the learned over-complete dictionary, $A=(a_1,a_2,\cdots,a_n)\in\mathbb{R}^{p\times n}$ is the sparse coding based on D, the sparseness is guaranteed by the l_1 norm of the matrix as a regularization term. It should be noted that each sub-dictionary should have the same number of dictionary atoms, in general, the number of sub-dictionary atoms is equal to the dimension of the data. Elastic Net is used to solve sparse coding coefficients:

$$A=\arg\min_A\frac{1}{2}\|X-DA\|_F^2+\lambda_1\sum_i\|a_i\|_1+\lambda_2\|A\|_F^2 \quad (1)$$

It can also be expressed as a sparse coding for a single sample x_i, namely:

$$a_i=\arg\min_{a_i}\frac{1}{2}\|x_i-Da_i\|_2^2+\lambda_1\|a_i\|_1+\lambda_2\|a_i\|_2^2 \quad (2)$$

where λ_1 and λ_2 are the regularization of the parameters used to balance these three terms. When set $\lambda_2=0$, the above formula is a l_1 sparse decomposition problem. It should be noted that, must set $\lambda_2>0$, to ensure that the objective function is differentiable.

B.Cost-Sensitive Dictionary Learning

In this paper, an optimal classification model is obtained according to the input feature for the minimum classification loss, and the cost of different misclassification is considered in the process of dictionary construction [22]. In the task of software defect prediction, we use Cost(+1) to represent the cost of misclassifying a defective module as defective-free module, and Cost(-1) represents the cost of misclassifying a defective-free module as defective module.

Using the iterative optimization sparse coding $a_i(x_i,D)$ to train the classifier parameter W, and using logistic loss as a classifier objective function. Therefore, the loss function can be defined as:

$$T(A(X,D),W)=\sum_{i=1}^n\ell(y_i,W,a_i(x_i,D)) \quad (3)$$

Where $\ell(y_i,W,a_i(x_i,D))=Cost(y_i)\log(1+e^{-y_iW^Ta_i(x_i,D)})$ is a weighting logic loss function, $a_i(x_i,D)$ is the sparse coding of sample x_i in dictionary D, $Cost(y_i)$ is the cost of misclassification.

Finally, the joint minimization objective function proposed in this paper can be expressed by the two-level optimization model [23]:

$$\begin{aligned}&\min_{D,W}T(A,W)+\frac{v}{2}\|W\|_2^2\\ &s.t.\quad A=\arg\min_A\frac{1}{2}\|X-DA\|_F^2+\lambda_1\sum_i\|a_i\|_1+\lambda_2\|A\|_F^2\end{aligned} \quad (4)$$

In the test phase, each test sample is sparsely encoded on the dictionary D according to the formula (2), and then the resulting sparse coefficients is used as the input of the previous optimal classifier. Finally, the test samples are classified according to the formula (4).

C. Optimization Algorithm

We constructing the projection first-order stochastic gradient descent algorithm (SGD) to solve the formula (4), using KSVD technology to initialize sub-dictionary for each class, as shown in algorithm 1. It consists of an outer SGD loop that increments the sample training data, each time the gradient of the classifier parameters W and dictionary D for sample is solved, and update W and D.

Algorithm 1 Stochastic Gradient Descent Algorithm (SGD) to solve formula (4)

Input: $(X,Y);v,\lambda_1,\lambda_2;D_0,W_0$ (the initial dictionary and classifier parameters); $Iter$ (the number of iterations); t_0,ρ (learning rate)

Output: D,W

1. For t=1 to $Iter$ do
2. Draw a subset (X_t,Y_t) from (X,Y);
3. Sparse coding: computer A^* using Feature -Sign algorithm:
   $A^*=\arg\min_A\frac{1}{2}\|X_t-DA\|_F^2+\lambda_1\sum_i\|a_i\|_1+\frac{\lambda_2}{2}\|A\|_F^2$;
4. Compute the active set: Λ (the nonzero suport of A);
5. Compute β^*:Set $\beta^*_{\Lambda^c}=0$ and
   $\beta^*_\Lambda=(D_\Lambda^TD_\Lambda+\lambda_2I)^{-1}\nabla_{A_\Lambda}[T(A,W)]$;
6. Choose the learning rate:
   $\rho_t=\min\left(\rho,\rho\left(t_0/t\right)\right)$;
7. Update D and W by a projected gradient step:
   $W=\prod_W\left[W-\rho_t(\nabla_WT(A,W)+vW)\right]$
   $D=\prod_D\left[D-\rho_t(-D\beta^*A^T+(X_t-DA)\beta^{*T})\right]$,
   where $\prod_W$ and $\prod_D$ are respectively orthogonal projections on the embedding space of W and D;

8. end for
9. return W and D.

In practice, the distribution of data is usually unknown, we sample in a random sorted training set. In addition, in order to allow each iteration to sample more samples rather than sampling only a single sample, this paper adopts the minibatch strategy. In order to get stable results in the experiment, 200 samples were sampled at each time.

IV. Experiment and Result Analysis

This section introduces the experimental setup in detail, including the dataset, the evaluation metric and the analysis of the experimental results.

A. Experimental Subject

Table I. NASA DATASET

dataset	No.attr	No.def	Defect modules
CM1	38	42	12.21%
JM1	21	1759	18.34%
KC1	21	325	15.54%
KC3	40	36	18.00%
MC2	40	44	34.65%
MW1	20	27	10.23%
PC1	20	61	8.04%
PC3	20	140	12.44%
PC4	38	178	12.72%
PC5	39	503	2.96%

The experimental data for this paper are based on 10 class-imbalanced data sets provided by NASA, as shown in table II. The NASA project is the software system or subsystem of NASA, which contains static code metrics for software modules and the corresponding defect labels (defective software modules are labeled as "Y", and defective-free module are labeled as "N"). The ratio of the class-imbalance, that is, the percentage of the number of minority classes and the number of majority classes ranges from 2.96% to 34.65%, the size of the data set namely the number of software modules is between 127~17001.

B. Performance Evaluation

Generally, the performance of the prediction model is evaluated synthetically by using recall, precision, type I error rate (Err1), type II error rate (Err2) and F1 value.

The metrics of model evaluation are defined as follows:

Recall: The ratio of the number of modules correctly predicted as defect to the number of real defective modules.

$$recall = \frac{TP}{TP + FN} \tag{5}$$

Precision: The ratio of the number of modules correctly predicted as defect to the number of modules predicted as defect.

$$precision = \frac{TP}{TP + FP} \tag{6}$$

Recall and precision are a contradictory measure, so it is not appropriate to use only one of them to evaluate the performance of the prediction model. The F1-measure can combine them to evaluate models, F1-measure is widely used in performance evaluation of prediction model and other software engineering research fields [12]. The calculation formula of F1-measure is the harmonic mean of recall and precision:

$$F1 = \frac{2 \times recall \times precision}{recall + precision} \tag{7}$$

Type I error rate (Err1): The ratio of the number of falsely predicted as defective modules and the number of defective-free modules in real.

$$Err_1 = \frac{FP}{TN + FP} \tag{8}$$

Type II error rate (Err2): The ratio of the number of falsely predicted as defective-free modules and the number of defective modules in real.

$$Err_2 = \frac{FN}{TP + FN} \tag{9}$$

From the above ratio relationship can be seen, Err1 is relative to the real defective-free module predict as defective module, and Err2 is relative to the real defective module predict as defective-free module, both of them should be used simultaneously. Due to the different costs of detection and maintenance of Err1 and Err2, the influence of the cost of misclassification on the model is defined by the Expect Cost of Misclassification (ECM) as shown in equation (10), where C1 and C2 are the cost of Err1 and Err2 respectively, corresponds to Cost (-1) and Cost (+1) in the cost matrix, Pndf and Pdf represent the prior probabilities of the defective-free modules and defective modules, respectively. In many cases, as the misclassification cost of a single class is difficult to obtain, the influence of different misclassification costs on prediction model is often analyzed by using the misclassification cost ratio instead of a single misclassification cost, as shown in equation (11).

$$ECM = C_1 Err_1 P_{ndf} + C_2 Err_2 P_{df} \tag{10}$$

$$NECM = Err_1 P_{ndf} + (C_2 / C_1) Err_2 P_{df} \tag{11}$$

C. Parameter Settings

Similar to previous work in literature [16], the parameters λ_1 and λ_2, regularization parameter v, and learning rate ρ of Elastic-net are determined by cross validation, we use heuristics to reduce the search space of these parameters. The specific process is as follows: Firstly, we need λ_2 >0 when we prove that the objective function is different, and the choice of small λ_2 value is a necessary condition for the convergence of the algorithm. However, the experimental results show that the better sparse representation can be obtained when setting λ_2 =0; Secondly, we can achieve better reconstruction results when the parameter λ_1 is near 0.025, therefore we test in the given range $\lambda_1 = 0.025 + 0.0125k(k \in \{-3, \cdots, 3\})$ to achieve the parameter λ_1; Thirdly, when there are a large number of training samples, the regularization parameter v can be set as an arbitrary small value, such as setting v =10^{-5}, without enough training samples, this parameter is set by cross validation; Finally, the maximum number of iterations is set to 500. In addition, similar to the literature [14], in order to emphasize the influence of risk costs on the prediction model, according to the actual situation of the project, we set different C_2/C_1 values of misclassification costs.

In all experiments, if not specified, CSDL parameter settings are obtained through 10-fold cross validation, so as to avoid over-learning. Although our experiments verify that the choice of these parameters can achieve better prediction results, the finer parameter adjustment can further improve the performance of the algorithm.

D. Experimental Design

We designed different experiments to evaluate the proposed CSDL algorithm, and compared it with the defect prediction algorithms which are proposed in recent years, including Cost-sensitive Discriminative Dictionary Learning (CDDL) [14], SVM [5], NB [6]，Coding based Ensemble Learning (CEL)[16] and Cost-Sensitive Boosting Neural Network (CSBNN)[17].

Experiment one. Comparison of experimental results. The experiment was used to explore the effectiveness of the proposed method, and verify the performance of the CSDL algorithm. It was running on the 10 NASA datasets, and compared with 5 defect predictive algorithms which were proposed in recent years. We try to follow the parameters of the algorithm in the original document.

Experiment two. The impact analysis of different misclassification costs rate, which explore the effect of predicted model performance under the different rate of misclassification costs. We set the misclassification costs rate of C2/C1 range from 1 to 10.

E. Experimental Results and Analysis

We compared the CSDL with SVM, NB, CEL, CSBNN and CDDL, using 10-fold cross validation on the 10 NASA datasets, and calculate 10 sets of evaluation indicators (the value of F1) respectively. We focus on the study of optimal performance of the algorithm corresponding to other two cost-sensitive algorithms CSBNN and CDDL, so it is not need to have the same misclassification costs among them. The results are shown in TABLE II. The last row of table shows the average F1 on all the experimental datasets, the best F1 are presented with boldface.

From Table II, we see that the average F1-measure of CSDL is the highest. Firstly, we compared CSDL with NB, SVM and CEL, CSDL can obtain the best F1-measure in all of the experimental data sets. For example, in the KC1 data set, the F1-measure of NB, SVM and CEL are 0.376, 0.295 and 0.352 respectively, while the F1-measure of CSDL is 0.446.

Secondly, we compared CSDL with two cost-sensitive algorithm CSBNN and CDDL. In order to obtain the optimal predict performance, we select different misclassification costs to different projects. Compared with CSBNN, among all of the NASA data sets, CSDL can obtain the better predict performance through adjustment corresponding misclassification cost rate. For example, in the PC3 data set, the optimal F1-measure of CSDL is 0.435, which is higher than CSBNN’s 0.382. While the F1-measure of CSDL are not all higher than CDDL. For example, in the JM1 data set, the optimal F1-measure of CSDL is 0.380, which is less than CDDL’s 0.395. In most cases, the F1-measure of CSDL is higher than CDDL.

In order to evaluate the performance of CSDL fairly and objectively, we make use of Wilcoxon signed rank sums test method to analysis the experimental results of F1-measure at a 5% significance level. That is, the performance difference between two comparison methods were statistically significant when p is less than 0.05. As shown in Table III, in all NASA data sets, only one value of p is larger than 0.05. Thus, the performance difference between CSDL and other five methods were statistically significant.

TABLE II. F1-MEASURE OF DIFFERENT METHODS

DATASET	NB	SVM	CEL	CSBNN	CDDL	CSDL
CM1	0.325	0.214	0.279	0.312	0.366	**0.398**
JM1	0.332	0.311	0.341	0.383	**0.395**	0.380
KC1	0.376	0.295	0.352	0.405	0.435	**0.446**
KC3	0.385	0.382	0.353	0.384	0.421	**0.455**
MC2	0.456	0.521	0.501	0.567	**0.632**	0.618
MW1	0.312	0.275	0.278	0.336	0.382	**0.395**
PC1	0.284	0.362	0.327	0.315	0.415	**0.421**
PC3	0.291	0.87	0.365	0.382	0.413	**0.435**
PC4	0.367	0.475	0.492	0.463	0.541	**0.577**
PC5	0.331	0.176	0.357	0.365	**0.600**	0.576
AVG.	0.346	0.330	0.364	0.391	0.460	**0.471**

TABLE III. WILCOXON RANK TEST P VALUE OF CSDL AND OTHER METHODS

DATASET	CSDL				
	NB	SVM	CEL	CSBNN	CDDL
CM1	2.95×10^{-9}	1.21×10^{-10}	3.71×10^{-5}	6.73×10^{-7}	2.91×10^{-7}
JM1	1.96×10^{-11}	3.92×10^{-6}	6.16×10^{-9}	0.1323	1.54×10^{-3}
KC1	3.66×10^{-9}	9.12×10^{-11}	5.74×10^{-7}	6.21×10^{-8}	9.61×10^{-6}
KC3	7.53×10^{-5}	5.71×10^{-3}	1.19×10^{-5}	3.26×10^{-6}	8.71×10^{-5}
MC2	6.68×10^{-8}	5.69×10^{-6}	3.21×10^{-6}	1.73×10^{-6}	9.91×10^{-8}
MW1	3.19×10^{-8}	2.91×10^{-5}	2.27×10^{-4}	1.88×10^{-5}	2.57×10^{-8}
PC1	1.31×10^{-4}	5.79×10^{-5}	3.29×10^{-6}	3.99×10^{-8}	5.41×10^{-7}
PC3	4.25×10^{-8}	2.73×10^{-10}	4.05×10^{-6}	4.26×10^{-5}	5.89×10^{-6}
PC4	9.37×10^{-11}	4.43×10^{-8}	4.75×10^{-8}	5.42×10^{-9}	4.19×10^{-7}
PC5	4.13×10^{-13}	4.28×10^{-7}	4.63×10^{-10}	4.63×10^{-11}	2.55×10^{-5}

We compared CSDL with other two cost-sensitive algorithms CSBNN and CDDL, and set different misclassification cost rate to analysis how it affects the module performance. We make use of evaluation index Err1, Err2 and NECM to describe the impact of misclassification cost rate on the overall cost. The performance of the model is evaluated synthetically by the F1-measure. Due to space limitations, only the experiment results of data sets KC1, CM1 and PC1 under the different misclassification cost rate are given, we set C_2/C_1 range from 1 to 10, as shown in Figure 2~5.

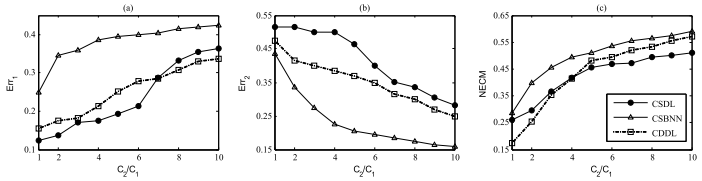


Figure 2. The performance comparison of KC1 dataset

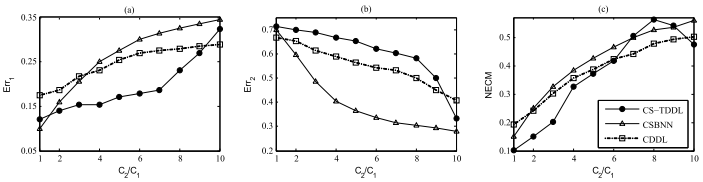


Figure 3. The performance comparison of CM1 dataset

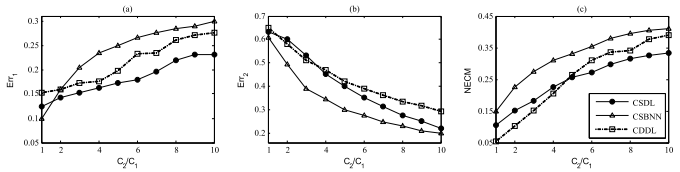


Figure 4. The performance comparison of PC1 dataset

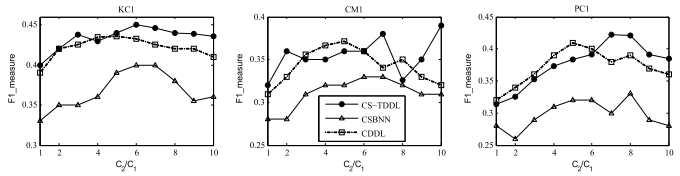


Figure 5. The F1-measure comparison on the KC1,CM1 and PC1

Figure 2~4 show that, with the increase in the misclassification cost rate, more and more defective modules were successfully detected, while the number of detective-free modules detect as defective modules also increases. The CDDL algorithm is more smooth in the Err1, Err2 indicators. CSDL and CDDL in the performance of NECM is always better than CSBNN algorithm. In the data sets KC1 and PC1, when the misclassification cost rate is less than 4, the performance of NECM on CSDL is not better than CDDL, but as the misclassification cost rate increase(larger than 5), the performance of NECM is significantly better than other methods. In the data set CM1, CSDL can get the best NECM performance when the misclassification cost rate is lower than 6, but the NECM performance of the CSDL algorithm decreases with the increase of the misclassification cost rate. In most cases, CSDL is always able to get the best NECM performance by effectively adjusting the misclassification cost rate.

In addition, the performance of three cost-sensitive algorithms on KC1, PC1, CM1 datasets is evaluated synthetically using F1-measure. As can be seen from Figure.5, the F1 performance of the prediction model changes significantly with the increase of the misclassification cost rate. Although in some cases, the F1 performance of CSDL is not optimal, but we can always make CSDL algorithm to achieve better F1 value by adjustment misclassification cost rate reasonably. In the actual application scenario, the software module's misclassification cost rate is difficult to obtain. As a result, it is often necessary to obtain an optimal predictive performance through an extensive search process for the misclassification cost rate.

V. CONCLUSIONS

In this paper, we propose a big data oriented cost-sensitive dictionary learning method to predict software defect, and use the logic regression classifier as the final classification judgment result. We iteratively optimize the dictionary atom and classifier parameters, and take full account of the different misclassification cost rate to improve the performance of the classifier. The experimental results show that our method has better detection effect than the other five methods, and can provide better support for software testing.

ACKNOWLEDGMENT

The research in this paper was partially supported by the National Natural Science Foundation of China under Projects No. 61373039, No. 61170022, No. 61003071 and No. 91118003.

REFERENCES

[1] G. Czibula, Z. Marian, and I. G. Czibula, “Software defect prediction using relational association rule mining,” Information Sciences, vol. 264, pp. 260–278, 2014.
[2] Pizzi N J. A fuzzy classifier approach to estimating software quality [J]. Information Sciences, 2013, 241: 1-11.
[3] Okutan A, Yildiz OT. Software defect prediction using Bayesian networks, Empir. Softw. Eng, 2014, 19(1):154-181
[4] Cheng M, Wu G, Wan H, et al. Exploiting Correlation Subspace to Predict Heterogeneous Cross-Project Defects[J]. International Journal of Software Engineering and Knowledge Engineering, 2016, 26(09n10): 1571-1580.
[5] Elish K, Elish M. Predicting Defect-prone Software Modules Using Support Vector Machines. Journal Systems and Software, 2008,81(5): 649-660
[6] Wang T, Li WH. Naïve Bayes Software Defect Prediction Model. Int. Conf. Computational Intelligence and Software Engineering. Wuhan: IEEE, 2010.1-4
[7] Zheng J. Cost-sensitive boosting neural networks for software defect prediction. Expert Systems with Applications,2010,37(6) : 4537–4543
[8] Zhou ZH, Liu XY. Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Trans. Knowledge and Data Engineering, 2006, 18(1):63–77
[9] Liu X Y, Wu J, Zhou Z H. Exploratory undersampling for class-imbalance learning[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2009, 39(2): 539-550.
[10] Xu Z, Liu Y, Mei L, et al. Semantic based representing and organizing surveillance big data using video structural description technology[J]. Journal of Systems and Software, 2015, 102: 217-225.
[11] Liu J, Yu X, Xu Z, et al. A cloud‐based taxi trace mining framework for smart city[J]. Software: Practice and Experience, 2016.
[12] Zhu G, He C, Shunxiang Z, et al. Weighted Indication-Based Similar Drug Sensing[J]. International Journal of Software Science and Computational Intelligence (IJSSCI), 2015, 7(1): 74-88.
[13] Wright J, Yang AY, Ganesh A, Sastry SS, Ma Y. Robust face recognition via sparse representation. IEEE Trans. Pattern Anal. Mach. Intell., 2009,31(2):210–227
[14] Jing XY, Ying S, Zhang ZW, Wu SS,Liu J. Dictionary learning based software defect prediction. In: Proc. of the 36th Int’l Conf. on Software Engineering. Hyderabad: ACM, 2014. 414–423
[15] Menzies T, Dekhtyar A, Distefano J,Greenwald J. Problems with precision: A response to comments on data mining static code attributes to learn defect predictors. IEEE Trans. Softw. Eng., 2007, 33(9):637–640
[16] Sun Z, Song Q, Zhu X. Using coding-based ensemble learning to improve software defect prediction. Systems, Man, and Cyb ernetics, Part C: Applications and Reviews, IEEE Transactions on,2012, 42(6):1806–1817
[17] Jiang Y, Cukic B, Menzies T. Cost curve evaluation of fault prediction models. IEEE Int. 19th International Symposium on Software Reliability Engineering.Seattle:IEEE,2008.197-206
[18] Mairal J, Bach F, Ponce J. Task-driven dictionary learning. IEEE Trans. Pattern Anal. Mach. Intell., 2012, 34(2):791–804
[19] Jiang Z, Lin Z, and Davis L. Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In: Proc. of the Int’l Conf. on CVPR,Barcelona:IEEE,2011.1697–1704
[20] Aharon M, Elad M, Bruckstein A. k-svd: An algorithm for designing overcomplete dictionaries for sparse representation. Signal Processing, IEEE Transactions on, 2006,54(11):4311–4322
[21] Zou H, Hastie T. Regularization and Variable Selection via the Elastic Net. J. Royal Statistical Soc. Series B, 2005,67(2): 301-320
[22] Cheng M, Wu G, Yuan M, et al. Semi-supervised Software Defect Prediction Using Task-Driven Dictionary Learning[J]. Chinese Journal of Electronics, 2016, 25(6): 1089-1096
[23] Colson B, Marcotte P, Savard G.An overview of bilevel optimization. Ann. Oper. Res., 2007, 153(1):235–256

Map-Reduce based Link Prediction for Large Scale Social Network

Ranjan Kumar Behera[1], Abhishek Sai Sukla[2], Sambit Mahapatra[3],Santanu Ku. Rath[4], Bibhudatta Sahoo[5], Swapan Bhattacharya[6]
Department of Computer Science and Engg.
National Institute of Technology, Rourkela, India.[1,2,3,4,5]
Jadavpur University, Kolkata, India.[6]
jranjanb.19@gmail.com[1], deva.abhi96@gmail.com[2], sambit9238@gmail.com[3],
rath.santanu@gmail.com[4], bdsahu@nitrkl.ac.in[5], bswapan2000@yahoo.co.in[6]

***Abstract*—Link prediction is an important research direction in the field of Social Network Analysis. The significance of this research area is crucial especially in the fields of network evolution analysis and recommender system in online social networks as well as e-commerce sites. This paper aims at predicting the hidden links that are likely to occur in near future. The possibility of formation of links is based on the similarity score between pair of nodes that are not yet connected in the social network. The similarity score, which we call link prediction score has been evaluated in Map-Reduce programming model. The proposed similarity score is based on both the structural information around the nodes and the degree of influence for neighboring nodes. The proposed algorithm is scalable in nature and performs quite well for large scale complex networks having good number of nodes and edges based on large pool of data or often termed as big-data. The efficiency and effectiveness of the algorithms are extensively tested and compared against traditional link prediction algorithms using three real world social network datasets.**

Keywords—Link Prediction; Preferential Attachment; Sim Rank; Jaccard Coefficient; Kartz Measure

I. INTRODUCTION

Online Social Networks(OSNs) are an inevitable part of today's society. Research on this area has given rise to an entirely new field of research called Social Network Analysis (SNA). The size of social network is observed to be increase in a very large scale during a short span of time; as a result, it demands a necessity for analysis of such a large sized network. However, traditional tools are found to be bit inefficient in analyzing large scale network. Distributed computing framework may be considered as an alternative for analyzing such large network in reasonable amount of time. Understanding the evolution of social network is one of the important aspects of social network analysis, where the aspect of "link prediction" plays an important role. Predicting links which may come up in a network in future can be utilized in different applications, such as recommender system in e-commerce sites, OSNs and predicting hidden links in a terrorist network etc. But, large-sized networks, which are quite prevalent these days, give rise to a need for considering a highly scalable method, rather than using other conventional methods with higher time and space complexity.

The evaluation of link prediction may have certain difficulties. First one, sparsity of large complex networks may lead to difficulty in designing a statistical model due to prior existence of very few link. Secondly, it is a difficult task to develop a highly efficient algorithm for big real-time networks. Generally, there exists a trade-off between computational time complexity and accuracy, because as, we go on increasing accuracy, time-complexity increases and vice-versa. Hence, the challenge is to design an accurate and efficient algorithm for analysis of huge and sparse networks. This paper presents an approach for prediction of links considering mutual neighbors between two nodes, their shortest distance and their influence in the network. This algorithm has been implemented using Map-Reduce technique making it suitable for large-scale social networks. By considering the small-world effect and scale-free network, this algorithm is observed to have less implementation complexity.

The subsequent sections of the paper are organized as follows: In section 2, the related work in the field of link prediction in large scale social network is discussed. Motivation behind the Map-Reduce approach for link prediction has been presented in section 3. In Section 4, the proposed algorithm has been presented. Section 5 indicates its implementation part. In section 6, a comparative study is presented by using the graphical representation. Section 6 concludes the paper and presents the scope for future work.

II. RELATED WORKS AND BACKGROUND DETAILS

The problem of Link-Prediction has been a trending topic of research for the past few years in the field of SNA. The first promising work in this field was carried out by Libebn-Nowell and Kleinberg [4], where authors introduced the significance of topological information and discussed as to how it can prove to be highly effective in predicting links in social networks. It can be observed from their work that topological information is highly effective in comparison with picking up random edges from a social network, as these networks are sparse in nature. There have been a lot of works based on proximity evaluation for link prediction in the paper [1] These methods seem to be not that much effective in large-scale networks. For example, escape-probability concept has been proposed as a powerful measure of direction-aware proximity by Zhou and Jia [6], which is closely related to rooted page rank. But the proposed

DOI reference number: 10.18293/SEKE2017-100

TABLE I: Link Prediction Algorithms

Link Prediction Algorithm	Equations	Description
Common Neighbor [1]	$LP_{CN}(x,y) = \|N(x) \cap N(y)\|$	Probability of having hidden links between two nodes increases with the number of common neighbors.
Preferential Attachment [2]	$LP_{PA}(x,y) = \|N(x)\| * \|N(y)\|$	Hidden links are most likely to be observed between higher degree nodes rather then smaller degree nodes.
Adamic Adar [3]	$LP_{AA}(x,y) = \sum_{z \in \{N(x) \cap N(y)\}} \frac{1}{log(N(z))}$	Adamic adar is the measure that gives higher preference to those pair of nodes which have neighbors that are not shared with other nodes.
Kartz Measure [4]	$LP_{KZ}(x,y) = \sum_{i=1}^{\infty} \beta^i \left\|path^i_{x,y}\right\|$	This measure is defined as the sum of all the paths (less than diameter) between two nodes.
Sim Rank [5]	$LP_{SR}(x,y) = \gamma \frac{\sum_{a \in N(x)} \sum_{b \in N(y)} LP_{SR(a,b)}}{\|N(x)\|\|N(y)\|}$	Probability of establishing link between two nodes is more if they are connected with more similar neighbors.

method of computing escape-probability helps only to scale networks with large number of nodes [7].

Some of the conventional techniques used for prediction of links are listed in Table I. In all of these techniques, a score is assigned to every possible, but not yet connected, pair of nodes (x, y), based on the input snapshot 'G' of the social network at a certain time. Link Prediction heuristics predict links between nodes based on their similarity i.e., more similar the nodes, higher is their score.

III. Motivation

In a social network, it is observed that an influential node tends to involve in network evolution rapidly as, more and more nodes try to associate themselves with this node. This aspect has been the intuition behind algorithms like preferential attachment and rooted page rank [8], but it does not consider either the shortest distance between the concerned nodes or the number of mutual neighbors, which these nodes share. Similarly, in certain heuristics like Jaccard, Adamic-Adar etc., mutual neighbors between two nodes are being considered without considering the influences of concerned nodes. To put a check on these limitations, a method has been proposed in this study, which considers not only the popularity of nodes in a network, but also the geodesic distance and mutual neighbors between concerned nodes. The significance of mutual neighbors and influence of mutual nodes are observed to be the important parameters which are being applied in other link prediction algorithms. In a social network, if two persons share popular friends then, there is a higher possibility of them getting connected, than the case where mutual friends are not that much popular. The motivation behind the use of eigenvector centrality is that it provides centrality score, which considers the importance of its neighboring nodes.

IV. Proposed Algorithm

The Proposed algorithm for link prediction is based on the Map-Reduce programming model, presented in Algorithm 1. In this algorithm, probable score for link has been measured only for those pair of nodes between which no edge exists.

Algorithm 1: Link Prediction using Map-Reduce(LPMR) Model

Input: The large scale social network $G = (V, E)$ in edge list format and eigen-vector centrality for each node in vector form.

Output: Link predicted score between i and j where A[i,j]=0 i.e. node pair between which no edge exists

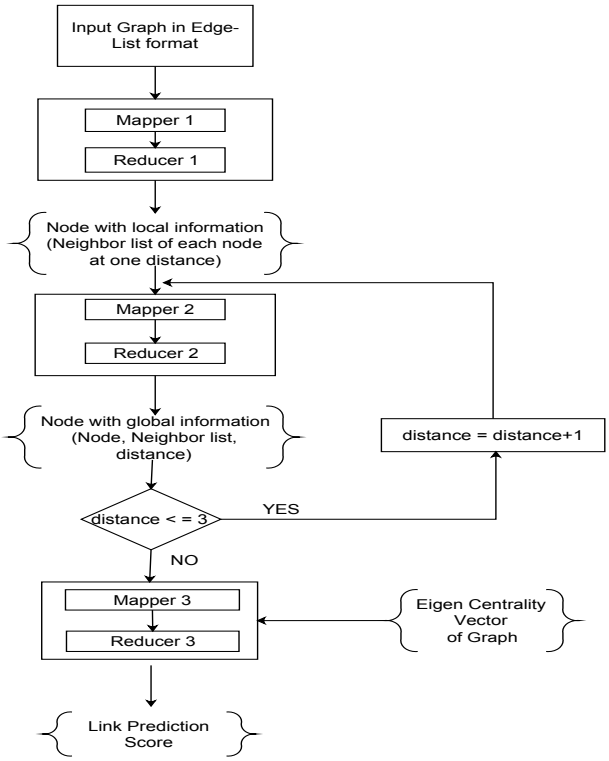


Fig. 1: Proposed Framework for Link Prediction

Step 1: The edge list is processed in first Map-Reduce phase. The first Map-Reduce phase is used to transform the edge list into adjacency list format, which contains local information of each node in the network. Local information may have list of neighboring nodes at one distance

Step 2: Modified adjacency list is then provided as input to the second Map-Reduce phase.

Step 3: Second Map-Reduce phase is used to apply extended ring search algorithm on modified adjacency list in order to get global information around the node.

Step 4: Global information contain structural information around a node upto path length four.

Step 5: Second Map-Reduce phase is repeated until the walk length is less than four. Neighbor list upto three walk length distance can be easily obtained in this step.

Step 6: The output of second Map-Reduce phase is then processed through Final Map-Reduce stage where link score

TABLE II: Real world datasets used for experiment

Datasets	No. of Nodes	No. of Edges	Diameter	Clustering Coefficient
Hamsterster Friendship [10]	1858	12534	14	0.901
Jazz Musician [11]	1133	2742	6	0.52
Ego Facebook [12]	2888	2981	9	0.0359

has been predicted.

Step 7: The link score has been predicted using the proposed similarity score which can be mathematically expressed as follow:

$$LS = \frac{1}{d}\left(EC(a) + EC(b)\right)\frac{\sum_{z\in\{N(a)\cap N(b)\}} EC(z)}{\sum_{k\in\{N(a)\cup N(b)\}} EC(k)} \quad (1)$$

where LS is the predicted link score between node a and b. EC(a) and EC(b) is the eigen-vector centrality score of node a and b respectively. d is the shortest distance between a and b.

The proposed LPMR model evaluates the similarity between nodes in the real world datasets. The initial input to the LPMR is the edgelist of graph G and eigen vector, where eigen vector centrality of all nodes has been provided. To compute shortest path of upto length L in graph G, customized form of expanding ring search algorithm is employed [9]. Traditional breadth first search (BFS) algorithm to find all shortest paths between any pairs of nodes has time complexity $O(n^3)$. Map-Reduce based algorithm has been applied for BFS in order to have global information of the node. Customized expanding ring search algorithm can find shortest paths of distance upto L from every node in a graph in a computational time complexity of $O(n)$, which is much less than the traditional algorithms. In this work execution time is further improved using Hadoop distributed frame work.

V. Implementation

A. Experimental Setup

Proposed Map-Reduce based Link prediction algorithms have been implemented on three real world social network datasets. All the experiments have been carried out on a cluster of 5 nodes, each with i7 processor having 3.4Ghz clock speed. Master node has the configuration with 1TB hard disk and 10GB RAM. It also acts as worker node. Each of other four nodes acts as slave or worker node. They all have symmetric configuration with 1TB hard disk and 20GB of RAM.

B. Dataset Used

The performance of the distributed link prediction algorithm i.e. LPMR is compared with the following existing link prediction algorithms:

a Common Neighbor (CN) [1]

b Kartz (KZ) Measure [4]

c Preferential Attachment (PA) [2]

d Adamic Adar(AA) [3]

e Sim Rank (SR) [5]

For testing our proposed algorithm, three real world social network datasets are considered. Details of the datasets are listed in TABLE II. The proposed algorithm use three pair of Map-Reduce components. Each component is intended for specific task of link prediction process. Input of the algorithm is the network with edge list format. As Map-Reduce takes only key-value pair as input, this format is to be strictly adhered in order to have compatibility with Map-Reduce model; however the dataset is transformed into a specific adjacency list format during different phases of Map-Reduce to gather the local information in the network. The first pair of Map-Reduce is used to discover the neighboring list of each node at one walk-length. After obtaining local information the dataset is then passed through second Map-Reduce phase where neighboring list of each node at four walk length has been discovered. This provides global information around the node in the network.

Link prediction score has been calculated in third Map-Reduce phase where eigen centrality of each node has been considered. Eigen centrality of each node has been provided at the beginning of the algorithm. The intuition behind using eigen centrality is that it captures the importance of each node by having connection with other important nodes. It can be observed that probability of establishing links between the node is high, if they are connected to more important nodes. The proposed link prediction score is also based on eigen centrality value of its common neighbors. It can be mathematically expressed as:

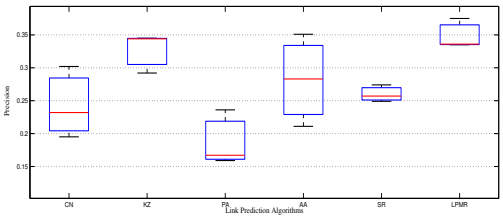
$$LS = \frac{1}{d}\left(EC(a) + EC(b)\right)\frac{\sum_{z\in\{N(a)\cap N(b)\}} EC(z)}{\sum_{k\in\{N(a)\cup N(b)\}} EC(k)} \quad (2)$$

where LS is the predicted link score between nodes a and b. EC(a) and EC(b) are the eigen-vector centrality scores of node a and b respectively. d is the shortest distance between a and b.

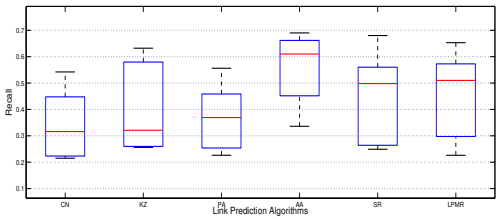
VI. Comparative Study

The performance of LPMR model is compared with other existing algorithms in terms of parameters such as precision, recall, f-measure and execution time by using three different datasets of social network. Few links have been randomly removed from the original datasets. After execution of link prediction algorithms, the predicted links have been compared with the links that have been removed prior to the experiments. All the experiments have been performed in Hadoop platform. The results have been aggregated and presented in the form of boxplot. Boxplot for Precision, recall, execution time and F-measure of link prediction probability is presented in Fig 2a, 2b, 2c and 2d respectively. Precision value for link prediction algorithms is presented in Table III. From Fig.2a, it can be observed that LPMR algorithm has better precision value as compared to those obtained using other traditional link prediction algorithms. Recall in information retrieval is the fraction of the links that are relevant to the hidden links being successfully retrieved. It is the number of detected links that actually exist before removal. From Fig. 2b, it can be identified that mean value of recall is better for Map-Reduce based algorithm.

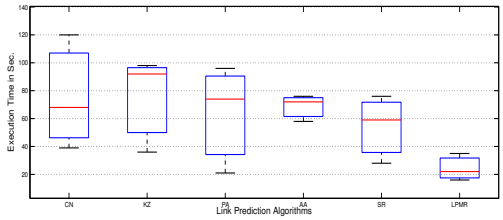
Execution time for the proposed algorithm i.e. LPMR is compared with other algorithms. The execution time (second)



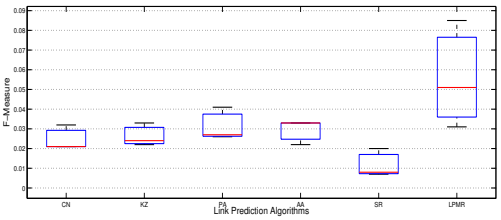
(a) Boxplot for Precision obtained in Different Algorithms



(b) Boxplot for Recall of Different Algorithms



(c) Boxplot for Execution time obtained in Different Algorithms



(d) Boxplot for F-Measure of Different Algorithms

Fig. 2: Comparative Analysis using Boxplot

TABLE III: Precision for different Algorithms

Network	CN	AA	Katz	SR	PA	LPMR
Hamsterster Friendship	0.232	0.211	0.292	0.249	0.167	0.385
Jazz Musician	0.195	0.283	0.345	0.257	0.159	0.336
Ego Facebook	0.302	0.351	0.344	0.033	0.274	0.375

TABLE IV: Execution Time for different Algorithms in Sec

Network	CN	AA	Katz	SR	PA	LPMR
Hamsterster Friendship	39	36	21	58	28	16
Jazz Musician	68	92	74	66	67	22
Ego Facebook	120	98	96	72	59	35

of different algorithms on different datasets are listed in Table IV. It can be observed from Fig. 2c that both minimum time and average time for the LPMR is comparatively less than other approaches. F-measure has been calculated for different algorithms on three datasets. F-measure value for Map-Reduce based algorithm is found to be more distributed as compared to other traditional algorithms. It can be observed from Fig.2d that mean value of F-Measure is more in LPMR algorithm .

VII. Conclusion and Future Work

Link prediction is one of the most important research directions in a number of application domains under SNA. It deals with revealing hidden links in the network. In this paper, an effort has been made in revealing hidden link in large scale network in distributed manner. Hadoop platform has been utilized for link prediction algorithm which is based on Map-Reduce programming model. The proposed algorithm has been extensively tested against few standard link prediction approaches. The Map-Reduce based Link prediction algorithm is found to be more suitable for processing large scale complex network. In Future, further enhancement to the proposed algorithm can be made for analyzing link prediction in dynamic network. Distributing processing tools like Storm and Spark can be implemented for streaming network where nodes and edges are added dynamically in continuous manner.

References

[1] Panpan Pei, Bo Liu, and Licheng Jiao. Link prediction in complex networks based on an information allocation index. *Physica A: Statistical Mechanics and its Applications*, 470:1–11, 2017.

[2] Benjamin Pachev and Benjamin Webb. Fast link prediction for large networks using spectral embedding. *arXiv preprint arXiv:1703.09693*, 2017.

[3] Eytan Adar and Lada A Adamic. Tracking information epidemics in blogspace. In *Proceedings of the 2005 IEEE/WIC/ACM international conference on web intelligence*, pages 207–214. IEEE Computer Society, 2005.

[4] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *journal of the Association for Information Science and Technology*, 58(7):1019–1031, 2007.

[5] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.

[6] Wen Zhou and Yifan Jia. Predicting links based on knowledge dissemination in complex network. *Physica A: Statistical Mechanics and its Applications*, 471:561–568, 2017.

[7] Pulipati Srilatha and Ramakrishnan Manjula. Similarity index based link prediction algorithms in social networks: A survey. *Journal of Telecommunications and Information Technology*, (2):87, 2016.

[8] Erjia Yan and Raf Guns. Predicting and recommending collaborations: An author-, institution-, and country-level analysis. *Journal of Informetrics*, 8(2):295–309, 2014.

[9] Jahan Hassan and Sanjay Jha. Optimising expanding ring search for multi-hop wireless networks. In *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*, volume 2, pages 1061–1065. IEEE, 2004.

[10] Hamsterster friendships network dataset – KONECT, October 2016.

[11] Jazz musicians network dataset – KONECT, October 2016.

[12] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.

A Practical Study on Quality Evaluation for Age Recognition Systems

Chuanqi Tao, ①② Hao Chen, ② Tiexin Wang ①, Jerry Gao ③, Wanzhi Wen ④

① College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

② School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China

③ Department of Computer Engineering , San Jose State University, USA

④ School of Computer Science and Technology, Nantong University

Correspondence to: taochuanqi@nuaa.edu.cn

Abstract—Face recognition system is a widely-used intelligent application nowadays. Existing recognition system evaluation methods primarily focus on recognition rate, i.e., the correct result. However, current research seldom focuses on the quality evaluation of face recognition systems. They seldom consider accuracy or the quality of recognition. To address this issue, this paper proposes several quality factors for evaluation. In addition, corresponding metrics for diverse quality factors are illustrated. Moreover, the paper presents an experimental study on a realistic non-trial face age recognition system using the proposed quality evaluation method. The study result shows the proposed method is feasible and effectiveness in quality evaluation.

Keywords: ***face recognition, age recognition application, quality factors, quality evaluation***

I. INTRODUCTION

Face recognition is one of the commonly-used intelligent systems in people's daily life. During the past ten years, face recognition technology has been widely focused and studied. Thus, face recognition systems are fast developing, from the feature points extraction or face matching in early years to face authentication, face age recognition and prediction and face dynamic tracing in recent times. Face recognition usually consists of the following steps: face image collecting, image preprocessing, image feature extraction, image processing (such as matching, recognition, and etc.). Facial age recognition has been concentrated and becomes a hot research issue recently in face recognition, which mainly realizes the judgment and prediction of age of a face picture. Some well-known age recognition systems have been used widely such as Microsoft's latest application--HOW-OLD and Alibaba cloud face age recognition API. When a user uploads photos, HOW-OLD will be able to recognize the gender and age of the given image. It works through learning and training of massive data in cloud, using detecting and identifying to tell the age of the image. HOW-OLD is mainly done by face detection, gender classification and age detection. Face detection is the basis of the other two technologies, and age detection and gender detection conduct the classification problem in the process of machine learning. This relates to the facial features of the portrait, the collection of learning data, the establishment of a classification model and model optimization. HOW-OLD analyzes the 27 points on people's faces to draw conclusions. These points are key nodes of the face, such as the pupil and the corner of the eye. Many researchers proposed optimize algorithm of facial age recognition. However, there are few quality evaluations for facial age recognition. The commonly-used metrics is recognition rate, which only tells the number of successful recognition. Age recognition system is a complex application that allows and tolerates errors (error is deviation from actual and expected value in software engineering). For example, assuming the real age of a face picture is 20 years old, if the recognition result is between 18 and 22, we may say that the system has a good recognition, rather than it must be recognized as 20-years-old exactly. Thus, how to measure the quality of face age recognition is an important issue for those kinds of systems.

Based on our recent research and survey, this paper proposes a set of reference quality factors for quality evaluation of face age recognition system. In addition, an experimental study is performed to discover the quality defects and problems of face age recognition. The remaining part of the paper is structured as follows. Section II is the related work. Section III presents the proposed quality factors and their measurements for facial age recognition systems. Section IV shows our experimental study and comparison with existing approach. Conclusions and future work are summarized in the end.

II. RELATED WORK

There are increasing quality problems resulting in erroneous testing costs in enterprises and businesses. According to IDC [1], the Big Data technology market will "grow at a 27% *compound annual growth rate* (CAGR) to $32.4 billion through 2017". In our previous work, the issue of quality assurance and validation for big data and applications was preliminary discussed [2, 3].

In the field of facial age recognition, most of the researchers focus on recognition algorithms. Du proposed a facial age estimation method based on sparsity constrained non-negative matrix factorization [4]. Yu proposed an age recognition method based on fusion error correcting output coding [5], which is a kind of SVM multi class classifier and integrates the error correcting output coding. Zhu proposed a 3D facial age recognition algorithm based on multi classifier fusion [6]. There are more proposed different age estimation

DOI reference number: 10.18293/SEKE2017-176

methods, such as [7-11].

Facial age recognition becomes a hot research domain in pattern recognition, so there is little evaluation method of facial age recognition. When evaluating the effect of the algorithms, researchers usually use the *average absolute error* to evaluate its quality [4, 7, 8, 9, 10], or the group recognition rate [5, 6, 8, 11]. The *average absolute error* is the average value of the absolute difference between real age and recognition age. The group recognition rate means the ratio of the occurrence when real age and recognition age are in the same age group.

Age recognition system not only need consider recognition rate or pass rate, but also the error between real age and recognition age. In the field of Agricultural, water conservancy, weather and economy, some error factors are used to evaluate the error, such as average absolute error, relative error [12-15]. But these quality factors haven't been used systematically and adaptively to evaluate an age recognition system. In Physics and Mathematics, there are also some error theory and error factors [16][17].

III. QUALITY FACTORS OF AGE RECOGNITION

Based on our investigation and analysis, we find that the *average absolute error* has some limitations in quality evaluation. For instance, when we recognize an image with 10-years-old in real as 20 years old, and recognize an image which is 70-years-old in real as 80 years old, the average absolute error is both 10 years old, but actually, recognize 10 as 20, the error rate is up to 100%, while recognizing 70 as 80, error rate is only 15%. Thus only using the *average absolute error* to evaluate the quality of facial age recognition is not reasonable. The *group recognition rate* used to evaluate age recognition is also not fully available, since the age recognition is a fault-tolerance application. When we define groups, there exists unavoidable boundary. For example, we define two group intervals as (10, 20] and (20, 30]. Both groups are not available in left boundary while available in right boundary. When the real age falls on the boundary just right, such as 20, if the system recognizes it as 19, then the result belongs to interval (10, 20]. However, when the system recognizes it as 21, it belongs to the interval (20, 30]. Thereby evaluation system may think there is an error. As we know, both 19 and 21 is an acceptable recognition result of real age 20.

Thus, only using average absolute error or the recognition rate cannot measure the quality of the age recognition system effectively. It is easy to ignore the hidden quality problems and is not conducive to the improvement of the system. The error between the facial age recognition results and actual real results, determines the accuracy of face recognition. Combing the error theory in theoretical physics and mathematics with the study of features of age recognition system, we propose several quality evaluation factors for age recognition systems shown in figure1.

The quality factors are mainly composed of two parts: recognition rate and accuracy rate. They are illustrated as follows.

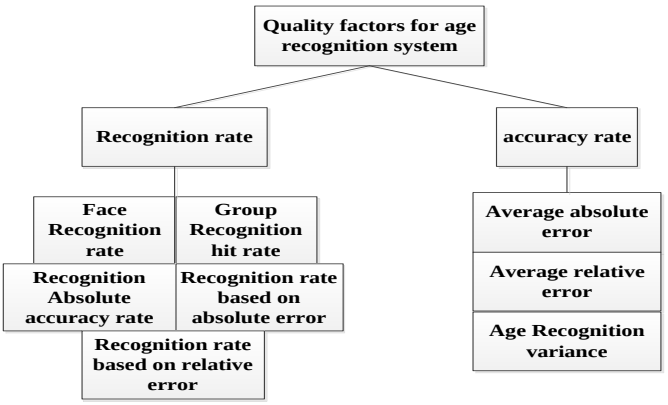


Figure 1 Quality factors for age recognition system

Recognition rate

The recognition rate refers to the ratio of recognition age to the actual age. The recognition rate is a reflection of the capability of age recognition such as the recognition ratio in different age groups with fault tolerance. Overall, the recognition rate reflects the correctness of an age identification system. Here we propose five quality calculating indexes for recognition rate. They are listed as: (1) the face recognition rate; (2) the group recognition hit rate; (3) Recognition absolute accuracy rate; (4) the recognition rate based on absolute error; and (5) the recognition rate based on relative error. The detailed calculation metrics for the five calculating indexes are explained as follows.

(1) Face recognition rate. Recognition rate is affected by a number of external factors, such as the angle of face, the size of face, and multi-faces in one image. There are more reasons that may cause the failure of face recognition in practical use. Here we adopt formula $R1 = NP/N$ to measure face recognition rate, where NP refers to the passed recognition cases (regardless of recognition accuracy), i.e., the system recognize a face and give its corresponding age, and N refers to the total cases that have faces without image quality problems.

(2) Group recognition hit rate. It refers to the ratio of the recognition cases that in the correct age group. For instance, we divide age range into several groups, with interval such as 10-20, 20-25, and etc. To avoid the boundary problem as discussed before, we assume if the real age is just at the boundary, both the left and right age group are considered to be correct for the recognition age. A good age recognition system should be able to make the recognition rate as high as possible. We use formula $R2 = NH/N$ to measure group recognition hit rate, where NH refers to the cases that hit the correct age group, and N refers to the total cases that pass the recognition system.

(3) Recognition absolute accuracy rate. It describes the ratio of the cases that the recognition age equals to the true age. In this metric, we use the formula $R3 = NC/N$, where NC means the cases that the recognition age is equal to the

DOI reference number: 10.18293/SEKE2017-176

actual age and N refers to the total cases that pass the recognition system.

(4) The recognition rate based on absolute error. It refers to that the error is divided with a fixed real number. The *recognition rate based on absolute error* represents the ratio of the cases that the error between real age and recognition age is smaller than the fixed real number as threshold. When the fixed real number threshold is zero, the value is the same as the index (3) above. The given formula is $R4 = COUNT(abs(age1 - age2) < e) / N$, where *age1* means the real age, *age2* is the recognition age, *e* represents the fixed real number threshold, and *N* refers to the total cases that pass the recognition system.

(5) The recognition rate based on relative error. This index shows that the error is divided based on the percentage of actual age. Since *recognition rate based on absolute error* cannot avoid the effect of the age size, we need to define new metrics for recognition rate. For instance, when we recognize an image with 70 years-old in true as 80 years-old, and recognize an image which is 20 years-old in true as 30 years-old, though the absolute error is both 10, it is obvious that recognizing 20 as 30 is more intolerable than recognizing 70 as 80, since the error for those two cases is 50% and 15% respectively. The *recognition rate based on relative error* is the ratio of the cases that the error between real age and recognition age is smaller than the percentage of actual age as threshold. The formula is given as $R5 = COUNT(abs(age1 - age2) / age1 < e) / N$, where *age1* means the real age, *age2* means the recognition age, e means the percentage, and N refers to the total cases that pass the recognition system.

Accuracy

Accuracy is a measurement of recognition error between recognition age and real age. If the 19-years-old is recognized as 20, from the recognition group rate, this age recognition is successful, because the error is small, but this does not mean there is no error. A good age recognition system should ensure a high recognition rate and a high accuracy with low error. The accuracy is divided into three indicators: *average absolute error*, *average relative error*, and *age recognition variance*. The explanations and their metrics are shown as follows.

- Average absolute error. It refers to the average absolute value of the error between real age and recognition age. The lower the value is, the closer the actual age and the age of recognition is, and the better the recognition quality is. The formula is given as $R5 = \sum abs(age1 - age2) / N$, where *age1* means the real age, *age2* means the recognition age, and N refers to the total cases that pass the recognition system.

- Average relative error. As explained in the quality factor #3 in recognition rate before, here we use average relative error to avoid the side-effect of age size. The formula is given as $R6 = \sum \frac{abs(age1 - age2)}{age1} / N$, where *age1* means the real age, *age2* means the recognition age, and *N* refers to the total cases that pass the recognition system.

- Age recognition variance. The absolute error and relative error of recognition cannot reflect the overall performance of the age recognition system. A big error may be evened out due to other small errors. Age recognition variance is more sensitive to the larger error point. The variance of the identification is defined as $R7 = \sum \frac{(age1 - age2) * (age1 - age2)}{N}$, where *age1* means the real age, *age2* means the recognition age, and N refers to the total cases that pass the recognition system.

IV. AN EXPERIMENTAL STUDY

In Section III, we discuss the quality factor and the calculation methods for quality evaluation of the age recognition system. In this section, we verify the effectiveness of the proposed quality factors and calculations by a realistic experimental study.

A. Study Object

The study object selects "Face Recognition Technology - Alibaba Cloud Computing Services Facial Age Recognition API" provided by Alibaba in china as the research object. The *base64* encoding of images is submitted to APIs, and the system returns with recognition result. The experiment data sets are selected from the wiki_crop.tar in the open face dataset IMDB-WIKI. There are total 52444 face data, and 10K images are selected randomly as experimental data sets.

B. Study Result Analysis

The study result of *age—cases* distribution is shown in figure 2, where the X axis represents age and the Y axis is the number of cases. The minimum age is 7 years old while the oldest is 98 years old, with 1 years old as the age interval. The actual age cases are marked in blue while recognition age cases are marked in orange.

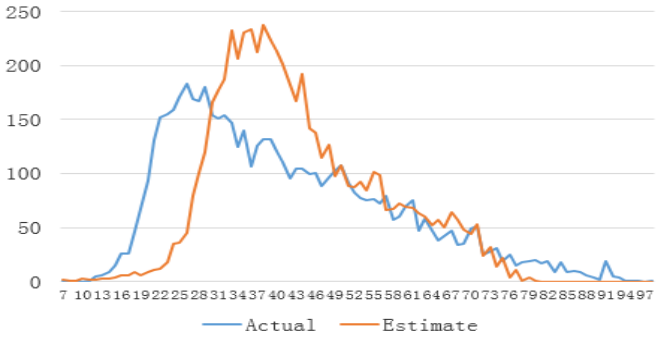


Figure 2 Age-Cases distribution

It can be seen from the figure that the actual age is primarily distributed in the age interval 22-37, and the recognition age is mainly distributed in the 30-40 age group, thus the peak of the age group is offset. Take the group of age

DOI reference number: 10.18293/SEKE2017-176

30-40 as an example, the number of age recognized as 30-40 is higher than the actual number. There are three possible reasons: (1) the system recognizes those under 30 years-old to 30-40; (2) the system recognizes those higher than 40 years-old to 30-40; and (3) it is also possible that the some of those belongs to 30-40 are recognized out of this range, but a number of people whose age is less than 30 or older than 40 are recognized as 30-40 age group, resulting the higher number.

The result of actual age—recognition age distribution is shown in Figure 3. The X-axis denotes the real age and the Y-axis denotes the average recognition age. The minimum age is 7 years old while the oldest was 98 years old with 1 year old as the age interval. The actual age cases are marked in blue while the recognition age cases are marked in orange. As shown in the Figure, the deviation is found as follows: Those between 0-30 years old are often recognized older than their actual age. Those between 70-90 years old in actual are often recognized younger. Age between 30-70 years old in actual are nearly the same as the recognition age. Overall, the error for cases in 0-30 and 70-90 groups is high while the error for cases in 30-70 groups is much lower.

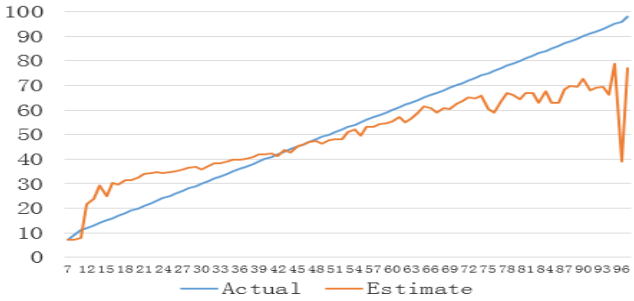


Figure 3 Actual age—Recognition age distribution

Now we analyze the result based on the proposed quality factors. The results of each quality factor that belongs to recognition rate are illustrated as follow.

(1) Face recognition rate. Among these 10000 image cases, 6081 cases are recognized as face images with given recognition age, and 3919 are recognized neither as face images nor be given recognition age. Through artificial identification, we find that the failed 3919 cases are divided into three categories: (a) image quality problems exist, as there is no face in the image, accounted for 33%; (b) though there do have faces in the image, the recognition face is too small, or face is profile, accounting for 54%, among which, profile faces account for 12% and small faces account for 42%; and (c) there exist images with front face are clear enough, accounting for 13%. In addition, among the passed 6081 cases, 5861 cases are recognized as single face while 220 cases are recognized as multi faces. Through our identification, we found that in the 220 multi-face images, there only 16% of the total images indeed have multi faces, and the other 84% only have single face while the system recognized it as multi-face. In summary, the number of cases that have faces without image quality problems is 6590, among which the face recognition rate is 88.92%. The study results show that the age recognition system cannot fully recognize. The possible reasons are as follows.

-The defect of age estimate algorithm;

-The amount of train data set that the age recognition provider use is not enough;

-The train data set that the age recognition provider use does not have universality or representativeness;

-The quality of the provided train data set has problems;

(2) Group recognition hit rate. Here we use the 6590 cases which passed the system. The result is shown in Figure 4.

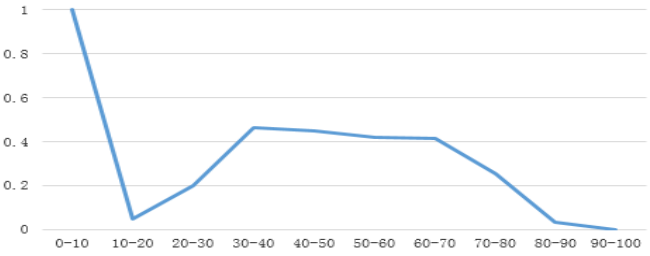


Figure 4 group recognition hit rate

From the figure we discover that the lowest recognition rate falls in the 10-20 age group and 80-100 age group while the higher recognition rate is in 30-70 group.

(3) Recognition absolute accuracy rate. As shown in Figure 5, the group with the lowest recognition accuracy is group 10-30 and group 80-100 while the higher recognition rate group is 30-70.

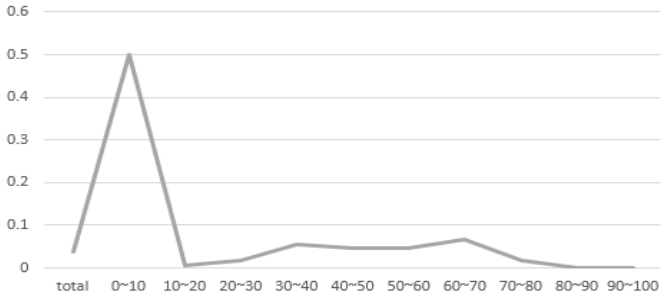


Figure 5 recognition absolute accuracy rate

(4) Recognition rate based on absolute error. The study result is shown in Figure 6.

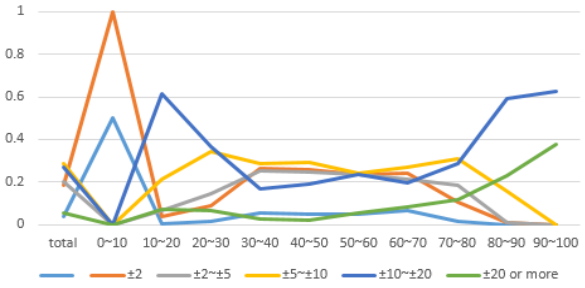


Figure 6 recognition rate based on absolute error map

From the figure we can see that, cases which have absolute error lower than 2 years and 2-5 years both account for 20%.

DOI reference number: 10.18293/SEKE2017-176

Cases with absolute error higher than 20 years account for 5%, and the others both account for 25%. In different age groups, absolute errors show differences. For instance, in the group of age 0-10, the errors are no more than 2 years; in the group of 30-60, the errors are usually between 0-5; and in the age group of 70-100, errors are usually much higher than other age groups.

(5) Recognition rate based on relative error. The result is shown in Figure 7. From the figure we can see that, cases which have relative error lower than 20% account for 45%. Cases which have relative error higher than 40% account for 20%. Others account for 10% or less. In different age groups, relative errors have differences. For example, in the group of 10-20, the errors are usually above 40%; in the group of 30-60, the errors are usually as low as 5%-15%. In the group of 70-80, though the absolute error is large, the relative error is small.

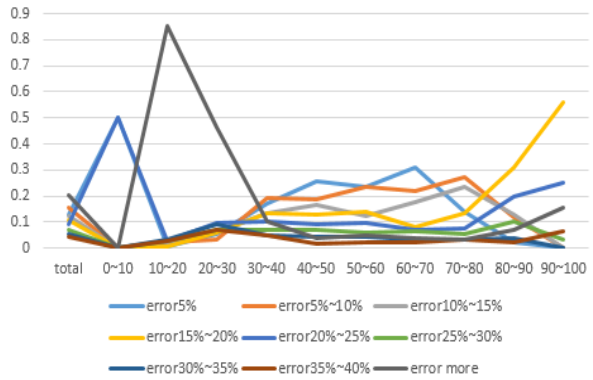


Figure 7 recognition rate based on relative error

The results of each quality factor of *recognition accuracy* are listed as follows.

(1) Average absolute error. The result is shown in Figure 8. We can see that the total average absolute error is about 9 years. Among these, the minimum average absolute error is at the section of 30-40 years, at about 6. The maximum average absolute error is in the group of 90-100. Therefore, the system works well on middle-aged person and works bad in the young and aged faces.

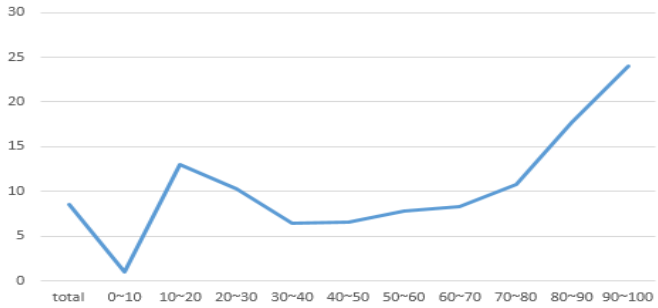


Figure 8. average absolute error map

(2) Average relative error. As shown in Figure 9, the total average relative error is about 25%. Among these, the average relative error in age group 40-60 is around 15% and the average relative error in group 10-20 is up to 75%.

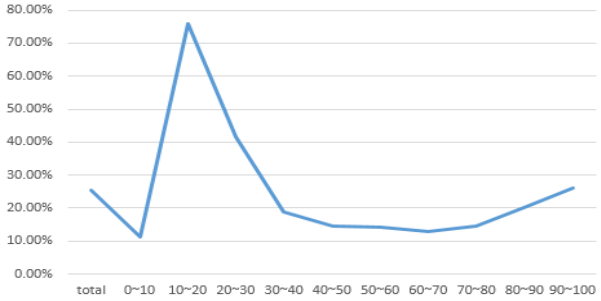


Figure 9 average relative error map

(3) Age recognition variance. As shown in Figure 10, variance can catch big errors instead of evening out them. We can see that the variance in group 10-20 and 70-100 are quite large, while the variance in the middle age is smaller.

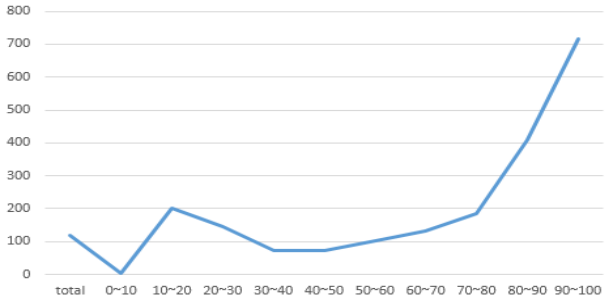


Figure 10 age recognition variance

C. Study Conclusions

Through the experimental study, we discover several defects of the chosen face age recognition system. They are listed as follows

(1) In some cases, the system cannot recognize some of the face images, let alone identify their age.

(2) The system recognizes some single people images as multi people images in some cases.

(3) If the face is too small, or face is profile, the recognition may fail sometimes.

(4) Regarding the recognition rate of absolute error, the ratio of those have errors less than 5 years (excluding completely correct identification) account for about 50%, thus the system has a good recognition ability.

(5) For recognition rate of relative error, the ratio of those have error larger than 40% account for too much and these usually occurs in the age group of 10-20 and 80-100. The system works poor in the young and the aged groups.

(6) In terms of absolute error, 30-50 years have the best performance, while the average absolute error of the aged is higher than 20%.

(7) According to the relative error, 30-50 year group shows the best performance, while the worst performance exists in the 10-20 age group.

D. Comparison with Other Evaluation Methods

DOI reference number: 10.18293/SEKE2017-176

Compared with *average absolute error*, such as the proposed methods in [4, 7, 8, 9, 10], to some extent, *average absolute error* can reflect whether the age recognition system shows good or bad. However, in the 10-20 age group of our experiment, the *absolute error* is not the highest. Nevertheless, in some cases, the *average absolute error* is more than 10 years. Compared with the actual age, the error is high up to more than 50% or even 100%. Therefore, the *average absolute error* is not enough to reflect the problem in this case.

Compared with the *average relative error* work- It can reflect the age recognition system is good or bad in some cases. In our study, the *average relative error* is not the highest in the 80-100 age group, and the *average relative error* of 10-20 and 20-30 groups are higher. However, there exist cases that the system recognizes a 90-year-old people to 70 or even younger. Only using this metric might lead to ignoring of the problems that the system works badly on the aged group.

Only using the recognition group hit rate, such as the methods in [5, 6, 8, 11], the *recognition group hit rate* to a certain extent reflects the performance of the age recognition system, However, it can only reflect the hit number, not the recognition quality and recognition accuracy. In addition, only using age group hit may ignore the error problems of those in the middle of an age group.

E. Study Limitations

There still existing some limitations of our study. They are listed below.

- The magnitude of experiment data is 10K. It can be expanded to millions or more in future work;
- The network or the net speed may affect the response time of the system;
- More quality factors and quality indicators can be proposed to evaluate the error between real age and estimate age;
- We do not take the quality of the picture into account. The noise or bad quality of the picture may affect the experiment result.

V. Conclusions and Future Work

This paper presents a practical study on a realistic recognition system based on the proposed quality evaluation method. The study results show that compared with the single metric such as *recognition rate* or *absolute error*, the proposed method performs better in finding quality issues existed in the age recognition system. In the future work, the quality factors in this paper can be extended to evaluate other intelligent systems such as prediction applications and recommendation applications.

References

[1] M. R. Wigan, R. Clarke. Big Data's Big Unintended Consequences[J]. Computer, 2013, 46(6):46-53.

[2] J. Gao, C. L. Xie, C.Q. Tao. Big Data Validation and Quality Assurance - Issuses, Challenges, and Needs[C], IEEE International Symposium on Service-Oriented System Engineering, 2016, pp 433-441.

[3] C.Q. Tao, J. Gao. Quality Assurance for Big Data Applications– Issues, Challenges, and Needs[C], the 28th International Conference on Software Engineering and Knowledge Engineering. 2016, pp 375-381.

[4] J.X. Du, Q Yu, Q.M. Qu. Facial age estimation method based on sparse constraint non negative matrix factorization[J]. Journal of Shandong University (SCIENCE EDITION), 2010, 45 (7): 65-69.

[5] M.S. Yu, A.Q. Zhu, X.M. Xie. Facial Age Recognition [J]. Co mputer and Modernization (in Chinese), 2013 (11): 210-213.

[6] Y. Zhu, J.L. Chang. 3 D Facial Age Recognition [J]. Computer Simulation, 2009, 26 (5): 248-250.

[7] X. Geng, Z.H. Zhou, K. Smithmiles. Automatic Age Estimation Based on Facial Aging Patterns [J]. IEEE Transaction on Pattern Analysis and Machine Intelligence, 2007, 30(2):368-368.

[8] Y. Dong, Y. Liu, S. Lian. Automatic Age Estimation Based on Deep Learning Algorithm [J]. Neurocomputing, 2015, 187:4-10.

[9] G. Guo,Y. Fu, C.R. Dyer, et al. Image-Based Human Age Estimation by Manifold Learning and Locally Adjusted Robust Regression[J]. IEEE Transactions on Image Processing, 2008, 17(7):1178.

[10] M. Ozaki, W. Motokawa. Dental Age Estimation by Two Computer Methods: Fuzzy Logic and Neural Network [J]. Biomedical Fuzzy & Human Sciences, 2000, 6:13-17.

[11] G. Guo, G. Mu, Y. Fu, et al. Human Age Estimation using Bio-inspired Features [C], IEEE Conference on Computer Vision and Pattern Recognition , 2009, pp 112-119.

[12] K. Kase, A. Makinouchi, T. Nakagawa. Shape Error Evaluation Method of Free-form Surfaces [J]. Computer-Aided Design, 1999, 31(8):495-505.

[13] T.S. Lee, E.E. Adam. Forecasting Error Evaluation in Material Requirements Planning (MRP) Production-Inventory Systems [J]. Management Science, 1986, 32(9):1186-1205.

[14] C. Lee, E. Choi. Bayes Error Evaluation of the Gaussian ML Classifier [J]. IEEE Transactions on Geoscience & Remote Sensing, 2000, 38(3):1471-1475.

[15] C. Cui, B. Li, F. Huang, et al. Genetic Algorithm-Based Form Error Evaluation [J]. Measurement Science & Technology, 2007, 18(7):1818.

[16] Q. Zhou. The Application of Markedness Theory in Error Evaluation: An Experimental Study [J]. Journal of Changchun University of Science & Technology, 2009.

[17] P. Jiang, J.Y. Zhao, J. Wei. Error Theory and Data Processing [M]. National Defence Industry Press, 2014.

Acknowledgement

This paper is supported by the National Natural Science Foundation of China under Grant No.61402229, 61502233, and 61602267; the Open Fund of the State Key Laboratory for Novel Software Technology (KFKT2015B10), the Postdoctoral Fund of Jiangsu Province under Grant No.1401043B, and the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant no. 15KJB52003.

DOI reference number: 10.18293/SEKE2017-176

FSCR:A Feature Selection Method for Software Defect Prediction

Xiao Yu[1,2,3], Ziyi Ma[2,3], Chuanxiang Ma[2,3*] ,Yi Gu[2,3],Ruiqi Liu[4], Yan Zhang[2,3]
[1]State Key Lab. of Software Engineering, Computer School, Wuhan University, Wuhan, China
[2]School of Computer Science and Information Engineering, HuBei University, Wuhan, China
[3]Educational Informationalization Engineering Research Center of HuBei Province, Wuhan, China
[4] International School of Software, Wuhan University, Wuhan, China
*Corresponding author email: mxc838@hubu.edu.cn

***Abstract*—Prediction the number of faults in software modules can be more helpful instead of predicting the modules being faulty or non-faulty. Some regression models have been used for predicting the number of faults. However, the software defect data may involve irrelevant and redundant module features, which will degrade the performance of these regression models. To address such issue, this paper proposes a feature selection method based on Feature Spectral Clustering and feature Ranking (FSCR) for the number of software faults prediction. First, FSCR groups the original features with spectral clustering according to the correlation between every two features. Second, FSCR employs ReliefF algorithm to compute the relevance between each feature with respect to the number of faults and selects top *p* most relevant features from each resulted cluster. We evaluate our proposed method on 6 widely-studied project datasets with four performance metrics. Comparison with five existing feature selection methods demonstrates that FSCR is effective in selecting features for the number of faults prediction.**

Keywords—software fault prediction;regression model;feature selection; spectral clustering

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 [2-5], 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 [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, prediction the number of faults in software modules can be more helpful instead of predicting the modules being faulty or non-faulty [6].

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 the number of faults prediction 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 most cases. In another similar study, Rathore et al. [12] presented an experimental study to evaluate and compare the other six regression algorithms for the number of faults prediction. The results found that decision tree regression, multilayer perceptron, and linear regression achieved better performance in many cases.

However, the performance of these regression models is still vulnerable to irrelevant and redundant module features that may undermine the prediction effect. It is crucial to apply feature selection to the number of faults prediction since feature selection can filter out irrelevant and redundant features by evaluating the contributions of module features. The output of feature selection is a subset of the original feature set. This feature subset is more effective for the number of faults prediction.

In this paper, we propose a novel feature selection method, FSCR, to support feature selection for the number of faults prediction. FSCR is short for feature selection based on Feature Spectral Clustering and feature Ranking, which enhances feature selection for the number of software faults prediction via a two-stage approach. First, FSCR groups the original features with spectral clustering according to the correlation between every two features. Second, FSCR employs ReliefF algorithm to compute the relevance between the features and the number of faults and selects top *p* most relevant features from each resulted cluster.

We evaluate our proposed feature selection method, FSCR, by answering two research questions on performance. Experiments are conducted on 6 publicly available projects.

DOI reference number: 10.18293/SEKE2017-081

Experimental results show that FSCR can effectively select features to improve the performance of the models for the number of faults prediction.

II. Related Work

In this section, we first briefly review the existing defect prediction methods. Then, we review the existing feature selection methods.

A. Defect Prediction

Many researchers have proposed various models for predicting the module being faulty or non-faulty. Support vector machine [13-14], neural networks [15], decision trees [16] and Bayesian methods [17] paved the way for classification-based methods in the flied of defect prediction. These methods used software metrics to properly predict whether a module is defect-prone or not.

A number of prior studies have investigated regression models on predicting the number of software faults. Graves et al. [18] presented a generalized linear regression based method for the number of faults prediction 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. 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 most cases. In another similar study, Rathore et al. [9] presented an experimental study to evaluate and compare the other 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 many cases. However, the prediction performance of these models gets worse when the defect datasets contain irrelevant and redundant features.

B. Feature Selection in Defect Prediction

A number of prior studies have investigated feature selection methods on predicting the module being faulty or non-faulty. Gao et al. [19] studied four different filter-based feature selection methods with five different classifiers on a large telecommunication system and found that the Kolmogorov-Smirnov method performed the best. Gao et al. [20] presented a comparative investigation to evaluate their proposed hybrid feature selection method, which first uses feature ranking to reduce the search space and then applies feature subset selection. In order to investigate different feature selection methods to classification-based bug prediction, Shivaji et al. [21] utilized six feature selection methods to iteratively remove irrelevant features until achieving the best performance of F-measure. Chen et al. [22] proposed a two-stage data preprocessing framework, TC, which combines feature selection and instance reduction. Liu et al. [23] proposed a new feature selection framework, FECAR, to conduct feature clustering and feature ranking.

III. Methodology

In this section, we present our FSCR method for the number of faults prediction. We first introduce the framework of our proposed method; then we present the detailed steps in the stage of feature spectral clustering and feature ranking.

A. The framework of our method

The method consists of two major stages: feature spectral clustering and feature ranking. Fig.1 illustrates the process of FSCR using a simple example.

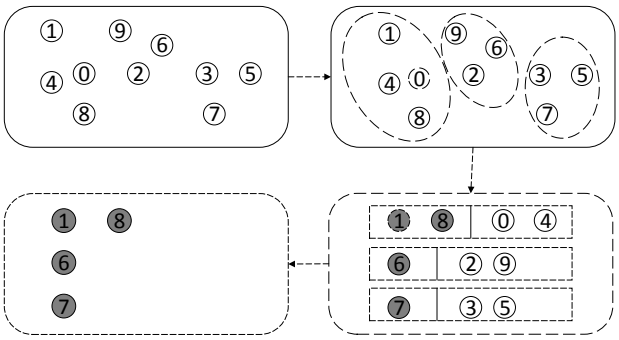


Figure 1. The process of FSCR

Assumes that the dataset has ten original features, represented by hollow circle in Fig.1. In the first stage, these features are partitioned into three clusters by using the spectral clustering algorithm, namely, C_1={0,1,4,8}, C_2={2,6,9} and C_3={3,5,7}. In the second stage, we rank all features in every clusters based on the relevance between each feature with respect to the number of software faults, and select the top p features from each cluster. Therefore, the final feature subset contains 1, 6, 7 and 8.

Therefore, the input of the FSCR method is the original feature set $\{f_1,f_2,\ldots,f_n\}$, the correlation measure FA between every two features, the relevance measure FB between each feature and the number of software faults, the number of the clusters k and the number of selected features m. The output of the FSCR method is the final feature subset R. The details are shown in the Algorithm 1.

B. The first stage

The first stage partitions the original features into k clusters such that features in the same cluster are similar and features in different clusters are dissimilar to each other. The main goal of the stage of feature clustering is to eliminate redundant features that have similar effect with other features. Note that in contrast to traditional clustering, our goal is to group features rather than instances.

This stage first uses the Pearson correlation coefficient to calculate the pairwise correlation between every two features using the following formula:

$$r=\frac{\sum_i(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_i(x_i-\bar{x})^2}\sqrt{\sum_i(y_i-\bar{y})^2}} \quad (1)$$

where values x_i and y_i denote the numeric values of the feature x and feature y in the i-th instance (i=1,2,…,n), $\bar{x}=\frac{1}{n}\sum_{k=1}^{n}x_i$ and $\bar{y}=\frac{1}{n}\sum_{k=1}^{n}y_i(i=1,2,\ldots,n)$.

Algorithm 1. FSCR method

Input:

Original feature set $\{f_1,f_2,\ldots,f_n\}$

Correlation measure FA between every two features

Relevance measure FB between each feature and the number of software faults

Number of the clusters k

Number of selected features m

Output:

Final feature subset R

/*The first stage: feature clustering*/

1: **for** i=1 to n **do**

2: **for** j=1 to n **do**

3: Compute the correlation between f_i and f_j using FA;

4: **end for**

5: **end for**

6: Partition original n features into k clusters $\{C_1,C_2,\ldots,C_k\}$using spectral clustering algorithm;

/*The second stage: feature ranking*/

7: **for** i=1 to n **do**

8: Using the relevance measure FB to compute the relevance between f_i and the number of the software faults;

9: **end for**

10: **for** i=1 to n **do**

11: Ranking the features in C_i in descending order according to the relevance;

12: **end for**

13: **for** i=1 to k **do**

14: Adding top $\left[\frac{|C_i|\times m}{n}\right]$ features of C_i into R;

15: **end for**

16: **return** R ;

Then, this stage uses spectral clustering to cluster the original feature set based on the correlation between every two features. Different from the other distance-based clustering algorithms, spectral clustering [24] makes use of the spectrum (eigenvalues) of the similarity matrix of the instances to perform dimensionality reduction before clustering in fewer dimensions. The similarity matrix can be defined as a symmetric matrix W, where W_{ij} represents a measure of the similarity between every two instances X_i and X_j.

C. The second stage

In this stage, we select top p relevant features from each resulted cluster to construct the final feature subsets. We first employ the ReliefF algorithm [25] to compute the relevance between each feature and the number of the software faults. ReliefF randomly selects an instance R_i, but then searches for k of its nearest neighbors from the same class, called nearest hits H_j, and also k nearest neighbors from each of the different classes, called nearest misses M_j(C). It updates the quality estimation for all features depending on their values for R_i, hits H_j and misses M_j(C). If instances R_i and H have different values of the attribute A then the attribute A separates two instances with the same class which is not desirable so we decrease the quality estimation W[A]. On the other hand if instances R_i and M have different values of the attribute A, then the attribute A separates two instances with different class values which is desirable so we increase the quality estimation W[A]. The whole process is repeated for q times, where q is a user-defined parameter. In this experiment, we use the default parameter specified by *sklearn* [26].

Then, we rank the features in C_i in descending order according to the relevance and select $\left[\frac{|C_i|\times m}{n}\right]$ features from each clusters, where $|C_i|$ is the number of the features in the cluster C_i, m is the size of the final feature subset and n represents the number of the original features. The selected features construct the final feature subset. According to literature [19], we select $\lceil \log_2 n \rceil$ features from the original features.

IV. Experiment Setup

A. Data set

In this experiment, we employ 6 available and commonly used software project datasets with their 22 releases which can be obtained from PROMISE [27]. The details about the datasets is shown in Table I, where *#Instance* represents the number of instances, *#Defects* represents the total number of faults in the release, *%Defect* represents the percentage of defect-prone instances, and *Max* is the maximum value of faults. There are the same 20 independent variables (the 20 feature metrics) and one dependent variable (the number of faults) in the six datasets. A comprehensive list of the metrics refers to literature [12].

TABLE I. Details of Experiment Dataset

Project	*Release*	*#Instance*	*#Defects*	*%Defects*	*Max*
Ant	Ant-1.3	125	33	16.0%	3
	Ant-1.4	178	47	22.5%	3
	Ant-1.5	293	35	10.9%	2
	Ant-1.6	351	184	26.2%	10
	Ant-1.7	745	338	22.3%	10
Camel	Camel-1.0	339	14	3.4%	2
	Camel-1.2	608	522	35.5%	28
	Camel-1.4	872	335	16.6%	17
	Camel-1.6	965	500	19.5%	28
Jedit	Jedit-3.2	272	382	33.1%	45
	Jedit-4.0	306	226	24.5%	23
	Jedit-4.1	312	217	25.3%	17
	Jedit-4.2	267	106	13.1%	10
	Jedit-4.3	492	12	2.2%	2

Project	*Release*	*#Instance*	*#Defects*	*%Defects*	*Max*
Synapse	Synapse-1.0	157	21	10.2%	4
	Synapse-1.1	222	99	27.0%	7
	Synapse-1.2	256	145	33.6%	9
Xalan	Xalan-2.4	724	111	15.3%	7
	Xalan-2.5	804	388	48.3%	9
	Xalan-2.6	886	412	46.5%	6
Xerces	Xerces-1.3	503	69	13.7%	30
	Xerces-1.4	589	438	74.4%	62

B. Performance measures

Since CERFS is a model to predict the number of faults, it should be evaluated using criteria for regression models. In the experiment, we employ root mean square error (RMSE) to measure the performance. In addition, considering the imbalanced characteristic of software defect datasets, we also employ three commonly used performance measures that evaluate classification models, including pd, pf and G-measure. These performance measures are defined in Table III and summarized as follows.

TABLE II. PERFORMANCE MEASURES

		Actual	
		yes	***no***
Predicted	***yes***	***TP***	***FP***
	no	***FN***	***TN***
pd	$\frac{TP}{TP+FN}$		
pf	$\frac{FP}{FP+TN}$		
G-measure	$\frac{2*pd*(1-pf)}{pd+(1-pf)}$		
RMSE	$\sqrt{\frac{\sum_{i=1}^{n}\lvert\bar{Y}_i - Y_i\rvert^2}{n}}$		

● Probability of detection or pd is the measure of defective modules that are correctly predicted within the defective class. The higher the pd, the fewer the false negative results.

● Probability of false alarm or pf is the measure of non-defective modules that are incorrectly predicted within the non-defective class. Unlike pd, the lower the pf value, the better the results.

● G-measure is a trade-off measure that balances the performance between pd and pf. A good prediction model should have high pd and low pf, and thus leading to a high G-measure.

● RMSE measures the deviation between the predicted value $\bar{Y}_i$ and the actual value Y_i. It is a good measure of accuracy to compare prediction errors of different regression models for a given variable, e.g., the number of faults.

C. Research Questions

Our evaluation answers two research questions.

RQ1. Does our proposed FSCR method perform better than state-of-the art feature selection methods in terms of predicting the modules being faulty or non-faulty?

This question validates the important criterion of defect prediction: the performance improvement in terms of pd, pf and G-measure (as defined in Section IV-B).

RQ2. Does our proposed FSCR approach perform better than state-of-the art feature selection methods in terms of the accuracy of predicting the number of the software faults?

This question validates the important criterion of the number of faults prediction: the performance improvement in terms of RMSE. (as defined in Section IV-B).

We compare our method with six classical feature selection methods in defect prediction:(1)Full, (2)Chi-Square [28], (3) Signal-to-Noise [29], (4)Information Gain [30], (5)Gain Ratio [31], and (6)FSCAR [23].

Full is the original feature subset. Compared to this method, we can study whether the FSCR can improve the performance of the number of faults prediction. Chi-Square (CS) and Signal-to-Noise (S2N) are statistic-based feature selection methods. Information Gain (IG) and Gain Ratio (GR) is the probability-based feature selection method. FECAR is a feature selection method combining feature ranking and feature clustering proposed by Liu et al [23]. FECAR first clusters features via k-medoids method and then select several representative features from each cluster.

D. Experiment Procedure

In experiments, we performed 10-fold cross validation when training classifiers on the selected features throughout this paper, to avoid any potential problem of overfitting particular training and test sets within a specific project. In 10-fold cross validation, a dataset is divided into 10 folds at random. Nine of the ten folds take turns to be used as the training set while the other fold is used as the test set. The training data are used to build a regression model; then the built model is evaluated on the test data. The above procedure is repeated 300 times (10 folds 30 independent runs) in total for each feature selection method to avoid sample bias. Then, the mean values of performance for all methods are calculated.

In order to compare the performance of feature selection methods, we employ three regression models in defect prediction, Bayesian Ridge Regression (BRR), Gradient Boosting Regression (GBR) and Linear Regression (LR). The reason we choose these regression models is that these models perform best in predicting the number of software faults [11-12].

V. EXPERIMENT RESULTS

In this section, we present the experiment results to answer our two research questions mentioned above.

A. RQ1

As mentioned in Sections IV-C, we compare our method FSCR with six feature selection methods. Table IV records the pd, pf and G-meausre of six datasets with six different feature selection methods on three regression models, BRR, GBR, LR. The column "Full" presents the training set without involving any feature selection method; W/D/L, short for Win/Draw/Loss, denotes the number of projects, on which FSCR performs better than, the same as, or worse than another method, in terms of G-measure.

As is shown in the Table III, FSCR performs better G-measure values than all the other methods. For BRR model, FSCR achieves the best average pd and G-measure value, but fails in the best pf value. For GBR model, FSCR can achieve the best pf and G-measure values. For LR model, FSCR achieves best values in terms of all the three measures. The Win/Draw/Loss values shows that, on three regression models, FSCR outperforms others on over half of projects in terms of all the three measures.

TABLE III. AVERAGE PERFORMANCE OF 6 PROJECTS WITH THREE REGRESSION MODEL ON PD, PF, AND G-MEASURE

Model	Metric	Full	FSCR	CS	GR	S2N	IG	FECAR
BRR	PD	0.512	**0.584**	0.514	0.521	0.556	0.548	0.579
	PF	0.236	0.169	0.182	0.171	**0.164**	0.165	0.170
	G	0.613	**0.668**	0.591	0.586	0.642	0.622	0.665
	W/D/L	4/0/2		5/0/1	4/0/2	5/0/1	4/0/2	4/0/2
GBR	PD	0.479	0.521	0.466	0.498	**0.535**	0.501	0.513
	PF	0.157	**0.121**	0.206	0.142	0.123	0.161	0.126
	G	0.610	**0.637**	0.585	0.613	0.633	0.609	0.631
	W/D/L	6/0/0		5/0/1	4/0/2	6/0/0	4/0/2	4/1/1
LR	PD	0.504	**0.591**	0.434	0.536	0.523	0.511	0.586
	PF	0.244	**0.152**	0.183	0.213	0.221	0.175	0.159
	G	0.604	**0.671**	0.565	0.625	0.609	0.649	0.668
	W/D/L	5/0/1		4/0/2	4/0/2	6/0/0	6/0/0	3/0/3

Fig. 2 shows the box-plots of G-measure values, with six methods for three regression models on 6 projects. For BRR model, the median value by FSCR is much higher than that by all the other methods. For GBR model, the median value by FSCR is higher than that by CS and S2N, while is similar with that by GR and IG, and is a little lower than that by FECAR. However, the maximum by FSCR is much higher than FECAR and all the other methods. For LR model, the median is similar with that by GR and FECAR, while is much higher than that by S2N and IG. In addition, the maximum by FSCR is much higher than that by all the other methods.

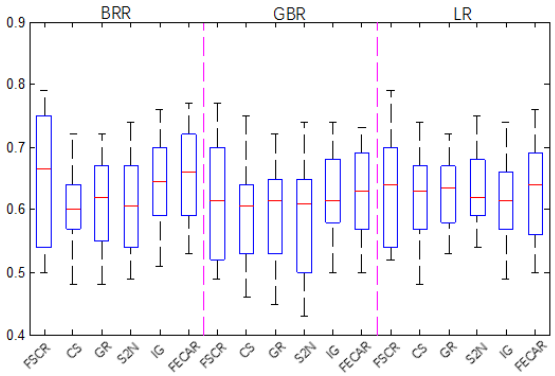


Figure 2. Box-plots for G-measure on 6 projects with three regression models.

RQ1 Summary. According to the experiment results in Table 4 and Figure 2, we conclude that FSCR can perform better than state-of-the-art feature selection methods in terms of predicting the modules being faulty or non-faulty.

B. RQ2

Tables IV, V and VI present the detailed RMSE values of each project on three regression models. From these tables, we can observe that FSCR performs better average RMSE value than all the other methods. The Win/Draw/Loss records also indicate that FSCR wins other methods on most projects on three regression models in term of RMSE measure. In addition, Hedges'g [32] is employed to demonstrate the effect size. The effect size of Hedges'g values are greater than 1.0 on most projects, which can be interpreted as a large improvement.

TABLE IV. RMSE VALUES ON 6 PROJECTS USING BAYESIAN RIDGE REGRESSION WITH THE HEDGES'G

Project	Full	FSCR	CS	GR	S2N	IG	FECAR
Ant	1.155	**0.829**	1.074	1.129	0.945	1.176	0.921
Camel	1.046	1.031	1.142	1.023	0.972	**0.824**	0.837
Jedit	1.426	0.986	0.965	0.965	1.028	0.975	**0.912**
Synapse	1.247	**0.892**	0.978	0.911	0.945	0.994	0.978
Xalan	1.010	**0.714**	1.123	1.101	0.897	1.109	0.925
Xerces	1.206	**0.821**	0.912	0.956	0.852	0.912	0.944
AVG	1.181	**0.878**	1.032	1.014	0.939	0.998	0.919
W/D/L	6/0/0		6/0/0	5/0/1	5/0/1	5/0/1	5/0/1
Hedges'g	**2.252**		**1.457**	**1.327**	0.657	0.980	0.462

TABLE V. RMSE VALUES ON 6 PROJECTS USING GRADIENT BOOSTING REGRESSION WITH THE HEDGES'G

Project	Full	FSCR	CS	GR	S2N	IG	FECAR
Ant	1.011	0.986	**0.894**	0.954	0.949	1.024	0.929
Camel	0.945	1.031	1.035	**0.927**	0.975	0.961	1.163
Jedit	1.295	**0.714**	0.917	1.082	1.021	0.913	1.075
Synapse	1.091	**0.821**	0.941	0.959	1.047	0.974	0.838
Xalan	0.906	**0.837**	1.109	1.056	1.145	1.127	0.914
Xerces	1.472	**0.892**	0.952	1.214	0.969	0.917	0.977
AVG	1.120	**0.880**	0.974	1.032	1.017	0.986	0.982
W/D/L	5/0/1		6/0/0	4/0/2	4/0/2	5/0/1	6/0/0
Hedges'g	**1.361**		0.939	**1.356**	**1.420**	**1.064**	0.873

TABLE VI. RMSE VALUES ON 6 PROJECTS USING LINEAR REGRESSION WITH THE HEDGES'G

Project	Full	FSCR	CS	GR	S2N	IG	FECAR
Ant	1.152	**0.957**	0.982	1.053	0.964	1.058	0.973
Camel	1.059	1.045	1.123	0.949	1.103	**0.934**	1.078
Jedit	0.914	**0.794**	0.994	1.027	1.025	0.853	0.935
Synapse	1.015	1.124	0.854	1.154	1.161	**0.927**	0.926
Xalan	1.205	**0.885**	0.942	1.048	1.054	1.185	0.910
Xerces	1.012	**0.921**	0.952	1.038	1.035	0.924	0.953
AVG	1.059	**0.954**	0.974	1.044	1.057	0.980	0.962
W/D/L	5/0/1		5/0/1	5/0/1	6/0/0	5/0/1	5/0/1
Hedges'g	0.944		0.193	**1.048**	**1.075**	0.219	0.086

RQ2 Summary. According to the experiment results in Tables 4-6, we conclude that FSCR can perform better than state-of-the-art feature selection methods in terms of the number of faults prediction.

VI. Conclusion and Future Work

In this paper, we propose a novel feature selection method for the number of faults prediction. The method involves the following two stages: in the first stage, we employ a feature spectral clustering method to cluster the original features; in the second stage, we select the highly relevant features from each cluster. Experiments on 6 project datasets indicate that the proposed method, FSCR, can perform competitive results for the number of faults prediction.

In the future, we will further investigate the impact of the parameters setting, such as the number of clusters and the number of features selected from each cluster. In addition, we would like to validate the generalization ability of our method on more datasets [33-34].

Acknowledgment

This work is partly supported by Educational Informationalization Engineering Research Center of HuBei Province.

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] Q. Song, Z. Jia, M. Shepperd, et al, A general software defect proneness prediction framework, Software Engineering, IEEE Transactions on, 2011, 37(3): 356-370.

[3] X. Yang, K. Tang, X. Yao, A Learning-to-Rank Approach to Software Defect Prediction, IEEE Transactions on Reliability, 2015,64(1): 234-246.

[4] R. Malhotra, A systematic review of machine learning techniques for software fault prediction, Applied Soft Computing, 2015, 27: 504-518.

[5] M. Shepperd, D. Bowes, T. Hall, Researcher bias: The use of machine learning in software defect prediction, IEEE Transactions on Software Engineering, 2014, 40(6): 603-616.

[6] N. E. Fenton, M. Neil, A critique of software defect prediction models, IEEE Transactions on software engineering, 1999, 25(5): 675-689.

[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] 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.

[14] 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.

[15] 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.

[16] 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.

[17] T. Wang, W. Li, Naive bayes software defect prediction model, Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on. IEEE, 2010: 1-4.

[18] 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.

[19] K. Gao, T. M. Khoshgoftaar, H. Wang. An empirical investigation of filter attribute selection techniques for software quality classification. Information Reuse & Integration, 2009. IRI'09. IEEE International Conference on. 272-277. IEEE, 2009.

[20] K. Gao, T.M. Khoshgoftaar, H. Wang, et al. Choosing software metrics for defect prediction: an investigation on feature selection techniques . Software Practice & Experience, 41(5):579-606, 2011.

[21] S. Shivaji, J. E. J. Whitehead, R. Akella, et al. Reducing features to improve bug prediction. Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering. IEEE Computer Society, 600-604, 2009.

[22] J. Chen, S. Liu, W. Liu, et al. A Two-Stage Data Preprocessing Approach for Software Defect prediction. Software Security and Reliability (SERE), 2014 Eighth International Conference on. 20 - 29. IEEE, 2014.

[23] S. Liu, X. Chen, W. Liu, et al. FECAR: A Feature Selection Framework for Software Defect Prediction. 2014 IEEE 38th Annual Computer Software and Applications Conference (COMPSAC). IEEE Computer Society, 426-435, 2014.

[24] Von Luxburg U. A tutorial on spectral clustering. Statistics and computing, 2007, 17(4): 395-416.

[25] Robnik-Šikonja M, Kononenko I. Theoretical and empirical analysis of ReliefF and RReliefF. Machine learning, 2003, 53(1-2): 23-69.

[26] http://scikit-learn.org/

[27] G. Boetticher, T. Menzies, T. Ostrand, The PROMISE Repository of Empirical Software Engineering Data, 2007 <http://promisedata.org/repository>.

[28] Jin X, Xu A, Bie R, et al. Machine learning techniques and chi-square feature selection for cancer classification using SAGE gene expression profiles. International Workshop on Data Mining for Biomedical Applications. Springer Berlin Heidelberg, 2006: 106-115.

[29] Plapous C, Marro C, Scalart P. Reliable A posteriori Signal-to-Noise Ratio features selection. 2005 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. IEEE, 2005: 66-69.

[30] Abadi D. Perbandingan Algoritme Feature selection information gain dan Symmetrical Uncertainty pada Data Ketahanan Pangan [J]. UT - Computer Science, 2013.

[31] Praveena P R, Valarmathi M L, Sivakumari S. Gain ratio patio based feature selection method for privacy preservation [J]. Ictact Journal on Soft Computing, 2011, 1(4).

[32] Kampenes, V. By, et al, A systematic review of effect size in software engineering experiments, Inform. Softw. Technol. 49.11 (2007) 1073-1086.

[33] Liu Z, Wei C, Ma Y, et al. UCOR: an unequally clustering-based hierarchical opportunistic routing protocol for WSNs, International Conference on Wireless Algorithms, Systems, and Applications. Springer Berlin Heidelberg, 2013: 175-185.

[34] Liu Z, Wei C, Ma Y, et al. UCOR: an unequally clustering-based hierarchical opportunistic routing protocol for WSNs, International Conference on Wireless Algorithms, Systems, and Applications. Springer Berlin Heidelberg, 2013: 175-185.

An empirical study on the influence of context in computing thresholds for Chidamber and Kemerer metrics

Leonardo C. Santos, Renata Saraiva, Mirko Perkusich, Hyggo O. Almeida and Angelo Perkusich
Federal University of Campina Grande, Campina Grande, Brazil
{leonardo.santos, renata.saraiva, mirko.perkusich, hyggo, perkusic}@embedded.ufcg.edu.br

Abstract

Software metrics have a fundamental role in the process of software quality management. However, in most cases, they are only used to quantify attributes, not supporting decision-making during the software life cycle. To support decision-making, it is necessary to give them by defining thresholds. In the literature, several approaches have been proposed with this purpose. On the other hand, most of them do not consider context factors such as the domain. Given this, in this paper, we evaluate if context factors influence the definition of thresholds for software metrics. Our work is restricted to Chidamber and Kemerer metrics, due to availability of data. We conducted an empirical study composed of two quasi-experiments. Each quasi-experiment uses an approach presented in the literature to define thresholds for software metrics, with the defined thresholds as the dependent variable. As the factor, we used a variable with two possible treatments: to consider the context or not. To define context, we used factors presented in the literature. As the objects of study, we used the source code of fifteen Java-based open-source projects. For measurement purposes, we used the six original Chidamber and Kemerer metrics. For both quasi-experiments, the accuracy of the definition of thresholds improved by considering the context. Therefore, we concluded that context factors influence the definition of the threshold for Chidamber and Kemerer metrics, which is an indicator that it influences other software metrics.

Software measurement; thresholds derivation; CK metrics.

1. Introduction

Software metrics is a collective term used to describe the wide range of activities concerning measurement in software engineering [1]. The reasons to use software metrics are: (i) to assist in project planning; (ii) to determine the strengths and weaknesses of the process and the product; and (iii) to evaluate the impact of a used particular technique. In practice, the use of metrics in large organizations such as HP, Motorola and NASA was evaluated by Ordonez and Haddad [2]. The result of this study indicated that metrics, when used early in the software development cycle, help to correct requirement failures and prevent errors.

Despite the potential benefits of metrics, in most cases, they are only used to quantify attributes and do not support decision-making [3]. According to Ferreira *et al.* [4], this has been identified as one of the reasons why metrics are not effectively used in industry. To maximize the use of metrics and support their interpretation, it is essential to define significant thresholds.

Many approaches to identify thresholds have been proposed in the literature. Ferreira *et al.* [4] used the *EasyFit* tool [5] to find the distribution most similar to the distribution of the metric. From percentile cuts, the categorization of thresholds was defined as good, moderate and bad. Foucault *et al.* [6] and Alves *et al.* [3] also used percentile cuts to compute thresholds. Foucault *et al.* [6] used a statistical analysis called *bootstrap* [7] to estimate the confidence interval of the percentile.

Receiver Operating Characteristic (ROC) curves were used by Shatnawi *et al.* [8] to associate metric to errors and find the threshold that provides better justification for these errors. In 2015, Shatnawi [9] proposed an approach that performs a logarithmic transformation on the data to reduce the skewness. In this approach, thresholds were extracted from the mean and standard deviation of the distribution related to the metric. Most of the works, such as Ferreira *et al.* [4], Alves *et al.* [3] and Shatnawi *et al.* [8], did not consider context variables to define the thresholds. On the other hand, Zhang *et al.* [10], in an empirical study that collected data from 320 software systems, demonstrated that the distribution of maintainability metrics values are influenced by context variables such as the application domain, the programming language and the number of changes made during the software development, giving indications that thresholds can also be influenced by these variables.

In this study, we hypothesize that context variables influence the software metrics' threshold, given that the distri-

DOI reference number: 10.18293/SEKE2017-044

bution of a metric influences its interpretation (i.e., thresholds). In this paper, we investigate if context variables influence the accuracy of the definition of software metrics' thresholds. With this purpose, we conducted an empirical study composed of two quasi-experiments. Each quasi-experiment uses a solution proposed in the literature to define thresholds for software metrics, with the calculated thresholds as the dependent variable.

Based on the number of referrals, we used the approaches proposed by Foucault *et al.* [6] and by Shatnawi [9]. As the factor of the study, we used a variable with two possible treatments: to consider the context or not. As the objects of study, we used the source code of fifteen Java-based open-source projects. For measurement purposes, we used the six original Chidamber and Kemerer (CK) metrics [11].

The remainder of this paper is organized as follows: in Section 2, we present the background regarding software metrics and thresholds derivation. In Section 3, we describe the empirical study design. In Section 4, we present the analysis and results. In Section 5, we present the threats to validity. Finally, in Section 6, we present our conclusions and future work.

2. Background

2.1. Chidamber and Kemerer Metrics

Metrics are used as a control instrument in the software development and maintenance process. Since 1970, hundreds of metrics have been proposed in the literature [12]. For instance, Chidamber and Kemerer [11] proposed a set of object-oriented metrics that is widely used by researchers, which are presented as follows:

- *Coupling Between Object classes (CBO)*: the coupling of a class is characterized by the number of relationships that a class has (i. e. counted for method calls, field accesses, inheritance, method arguments, return types, and exceptions). High coupling indicates a low reuse and a rise on the sensitivity to changes;
- *Depth of Inheritance Tree (DIT):* this metric calculates the depth of the inheritance tree (i.e., the distance between a class and its root class). It indicates that the deeper a class is in a class hierarchy, the more methods are inherited. Thus, the class becomes more complex and prone to errors;
- *Number Of Children (NOC):* represents the number of subclasses that inherit characteristics from a given class. This information provides evidence of the importance of the class importance to the project. A high value for this metric may correspond to inappropriate abstractions or mistakes related to inheritance concepts;
- *Response For a Class (RFC):* refers to the number of methods that can be executed by a class instance in response to an event or a received message. The higher the value, the greater the complexity of its testing and maintenance;
- *Lack of Cohesion in Methods (LCOM):* refers to the number of pairs of methods of a particular class in which the similarity is zero, minus the number of pairs of methods in which the similarity is nonzero. The similarity is calculated by the common use of variables of a class instance. Thus, a high value means that the class is not cohesive.
- *Weighted Methods per Class (WMC):* refers of the sum of the complexities of the methods of a given class. The higher the value, the more complex the class is.

Many studies have verified the relationship between CK metrics and faults in classes. In a systematic review, Jureczko and Madeyski [13] analyzed their effects on fault proneness. They showed that object-oriented metrics are better at finding fault than procedural metrics. Furthermore, they claim that CK metrics form the most common set of metrics to predict failures in classes. In Table 1, we detail the effect of CK metrics in a class' fault-proneness as presented in Shatnawi [9]. The first column corresponds to CK metrics. Second and third columns correspond to the quantity of research papers that present the negative and positive impact of CK metrics in a class' fault-proneness. Finally, the last column corresponds to the research papers that present data refuting the hypothesis that there is a relationship between CK metrics and faults.

Table 1: A summary of the CK metrics' impact in fault proneness [13, 9].

Metric	Positive	Negative	Not significant
WMC	12	0	0
DIT	4	2	6
NOC	2	3	4
CBO	11	0	1
RFC	11	0	0
LCOM	6	0	1

2.2. Thresholds Definition

The effective use of software metrics is hampered by the lack of significant thresholds [3]. In the literature, few metrics have defined thresholds. Furthermore, many researchers have proposed different approaches to define them [3, 4, 6, 14, 15, 9, 16, 8, 10].

Alves *et al.* [3] present a method that determines threshold empirically from measurement data. (i.e., benchmarking). The method is based on statistical properties of the

metric such as scale and distribution. To evaluate their approach, they collected data from 100 object-oriented software systems to calculate thresholds, which were successfully used to assist on software analysis, benchmarking and certification. The main risk of such a solution is to use thresholds to assist decision-making that were calculated for a different context.

In Sánchez-González *et al.* [16], an empirical study was performed to evaluate the effectiveness of two threshold definition techniques: ROC curves [8] and the Bender method [17] to define thresholds. As objects of study, they used measures for business process models. They concluded that ROC curves obtain more accurate thresholds.

In the works of Oliveira *et al.* [15, 14], the concept of relative thresholds is proposed as well as a tool for extracting these thresholds. Their approach handles the heavy-tailored distribution of source code metrics by complementing absolute thresholds with a percentage of software code entities that must follow it. The technique is validated with an industrial case study. As Alves *et al.* [3], its limitation is that the calculated threshold and percentage might be dependent on the context.

In Foucault *et al.* [6], a solution based on statistical methods was presented. This approach is based on (i) *double sampling* [18] to randomly selects projects samples; and (ii) *bootstrap* to estimate the thresholds based on quartiles. Despite the potential of this approach, the validation process was limited to a test to identify the best configuration for the approach itself since, according to the authors, the two statistical methods are widely used.

In Shatnawi [9], a solution based on logarithmic transformation was presented. In this approach, initially, the data is transformed using the natural log, leaving the symmetric data thus closer to a normal distribution. Afterward, a temporary reference value (T') is collected using the mean (M) and standard deviation (SD) so that $T' = M + SD$ or $T' = M - SD$. Finally, the T' is converted to the original distribution by using the exponent function of T', generating the final reference value.

3. Study Design

To evaluate if context factors influence the definition of software quality metrics' thresholds, we performed an empirical study composed of two quasi-experiments. Each quasi-experiment used one solution to define the thresholds of software metrics presented in the literature. The solutions used are the ones presented by Shatnawi [9] and Foucault *et al.* [6]. The defined thresholds are the dependent variable. As the factor, we used a variable with two possible treatments: to consider the context or not. The context factors were defined according to Zhang *et al.* [10]. As database, we used six CK metrics extracted of the source code of fifteen Java-based open-source projects.

3.1. Scope

The goal of the study is to evaluate if context factors influence the definition of software quality metrics' threshold in the context of Chidamber and Kemerer metrics. Therefore, we addressed the following research question:

RQ: Does considering context factors improve the quality of the definition of software quality metrics' thresholds?

Given the research question, we defined the following informal hypotheses:

H0: The results are the same or worse.

HA: The results are better.

3.2. Objects of study

For both quasi-experiments, we used the same data as Shatnawi [9], which is composed of several releases of fifteen open-source projects written in the Java programming language. We classified each project according to its application domain and the number of changes given the definitions presented in Zhang *et al.* [10]. The application domain of 73% of the projects is software development, 13% is the development of build tools, and 13% is the development of frameworks. For the number of changes, we used the thresholds defined by Zhang *et al.* [10] given the number of commits to the repository: 12 as *very small*; 123 as *small*; 413 as *medium*; 1142 as *large*; and 94853 as *extra large*.

3.3. Variables and treatment

In this study, we have two independent variables: (i) the technique used to define the thresholds and (ii) the context usage. As already stated, for the first variable (i), we used two options: the solutions presented in Shatnawi [9] and Foucault *et al.* [6]. In this study, we executed one quasi-experiment for each possible value of the first variable (i). For the second variable (ii), there are two options: to consider the context or not. To define the context, we used two out of three variables identified by Zhang *et al.* [10]: application domain and the number of changes, since the programming language of the base used is the same. Therefore, for each quasi-experiment, the treatment factor is the *context usage*. Furthermore, for each quasi-experiment, we have one dependent (*i.e.*, response) variable: the quality of the thresholds defined.

3.4. Measurement procedure

To evaluate the hypotheses, assess the research questions and research goals, we collected metrics. Furthermore, by defining the metrics, we formalized the hypotheses presented in Section 3.1 statistically test them. As stated in Section 3.3, for each quasi-experiment, we have one dependent variable: the quality of the thresholds defined. They were collected through the open source tool ckjm[1] and pub-

[1] http://www.spinellis.gr/sw/ckjm/

lished by the Metrics Repository [19, 20]. The measurement procedure was based on Shatnawi [9].

In this study, we assumed that the highest the source code quality, the fewer faults the software has. Since all of the objects of study are software written in Java code (*i.e.*, Object-oriented), as presented in 3.2, in order to measure the source code quality, we used the most popular Object-oriented metric set: CK metrics [11], which we presented in an overview in Section 2.1.

To measure the number of software faults, we used the BugInfo tool[2]. With this tool, we collected the number of faults of each class for all of the versions of the projects. This tool analyses the commits of a given project and detects, using regular expressions, if given classes had faults. Therefore, whenever a commit message is compatible with a defined regular expression, the number of faults for the given classes is incremented. Given this, we used the number of faults detected using the BugInfo tool to indicate source code quality.

To measure the threshold quality, we used the F-measure. For this purpose, the first step was to group the objects of study given the context variables presented in Section 3.3. Afterward, we separated the group into two samples. The first sample, called *context sample*, was composed of data from projects with the given characteristics: developed in *Java*, with *software development* as the application domain and classified as *extra large* regarding the number of changes. The second sample, called *control sample*, is composed of the remaining data.

For each quasi-experiment, we used the samples to compare the results found by applying the given thresholds' definition solution (i.e., Shatnawi [9] or Foucault *et al.* [6]) considering or not context variables. For each sample, we applied the k-fold cross-validation with $k = 10$. For every iteration, we used 9k to define the thresholds for six CK metrics: CBO, DIT, NOC, RFC, LCOM and WCM. We used 1k interaction' to evaluate the thresholds. As a result, we have two possible outcomes: (i) faulty classes, if $M \geq R$; or (ii) nonfaulty classes, if $M < R$, where M is the collected metric value and R is the defined threshold. The procedure is presented in Figure 1.

Table 2: The confusion matrix based on a threshold value.

Predicted	**Faulty**	**Nonfaulty**
M >= R	True positive	False positive
M < R	False negative	True negative

We used the confusion matrix presented in Table 2 to measure the performance of using the thresholds model in identifying actual fault classes using three measures: *Recall*, *Precision*, and *F-measure*. These measures are calculated as follows:

[2]https://kenai.com/projects/buginfo

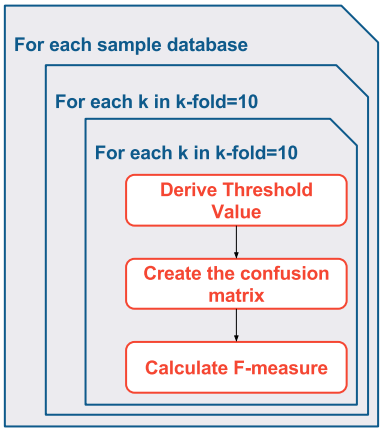


Figure 1: Design.

$$Recall = \frac{TP}{TP + FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$F - measure = \frac{(\beta^2 + 1) * Precision * Recall}{\beta^2 * Precision + Recall} \quad (3)$$

- The term β is used to assign a weight to *Recall*. In our work, β is equal to 1, and *Recall* and *Precision* are equally weighted.
- True Positive (TP): faulty classes that are correctly classified as such (*i.e.* there are faults fixed in the class and the metric value exceeds the threshold).
- False Negative (FN): faulty classes that are misclassified as nonfaulty (*i.e.* there are faults fixed in the class, but the metric value is less than the threshold).
- True Negatives (TN): nonfaulty classes that are correctly classified as such (*i.e.* there are no faults fixed in the class and the metric value is less than the threshold).
- False Positives (FP): nonfaulty classes that are misclassified as faulty (*i.e.* there are no faults fixed in the class, but the metric value exceeds the threshold).

The values of both *Recall* and *Precision* are between $[0, 1]$. Values that are closer to 1 mean better results, with 1 as an ideal value (*i.e.*, without FN or FP). According to Shatnawi [9], in practice, it is hard to achieve high *Recall* and high *Precision*. Finally, we used the *F-measure* to evaluate the overall performance of classification, combining *Recall* and *Precision*.

At the end, for each metric, we had a set of *F-measures* for the context sample and a different set for the control sample. We compared the pairs of *F-measures* of both sets using the non-parametric Wilcoxon test [21]. Thus, for each quasi-experiment, we formally defined a null and alternative hypothesis:

Therefore, for each quasi-experiment, we formally defined a null and alternative hypothesis:

H0: $\Omega \leq \Psi$, where Ω is the F-measure for the *context sample* and Ψ is the F-measure for the *control sample*.

HA: $\Omega > \Psi$.

4. Analysis and Results

4.1. First quasi-experiment: Foucault *et al.* approach

For the first quasi-experiment, we evaluated the solution proposed by Foucault *et al.* [6]. This approach, as presented in Section 2.2, is based on quantile analysis. We decided to use only the first percentile (80%) since it has the ability to represent the whole. An average of the thresholds' values found can be seen in Table 3. By comparing the thresholds defined for both samples (*i.e.*, context and control), it is possible to see that the DIT metric was the same for both samples.

Table 3: An average of the thresholds' values found using Foucault *et al.*'s approach.

Metric	Context threshold	Control threshold
WMC	9.00	10.05
NOC	0.00	0.20
CBO	9.05	10.10
RFC	25.15	27.85
DIT	2.50	2.50
LCOM	33.30	44.64

In Table 4, we present the average *F-measure* of the two data samples. Furthermore, we present the improvement of the threshold definition, $\theta = \Omega/\Psi$, and the *p-value* resulting from the Wilcoxon test. By analyzing the results, we conclude that, for all metrics, the threshold definition was slightly improved by considering the context. Since for all metrics $p-value < 0.05$, we refute the null hypothesis that states that considering the context to define the thresholds does not influence its definition for the solution presented by Foucault *et al.* [6].

4.2. Second quasi-experiment: Shatnawi approach

For the second quasi-experiment, we evaluated the solution proposed by Shatnawi [9]. This approach, as presented in Section 2.2, is based on log transformation.

Initially, for each k-fold interaction, two thresholds derived from each of the six CK metrics, one for each of the two database samples. An average of the thresholds' values found can be seen in Table 5. By comparing the thresholds for both samples, the differences were minimal.

Table 4: Results for Foucault *et al.*'s approach.

Metric	F-measure		θ	*p-value*
	Context	Control		
WMC	0.23	0.20	15.00%	1.62e-04
NOC	0.42	0.18	133.30%	5.41e-06
CBO	0.23	0.14	64.30%	5.41e-06
RFC	0.22	0.19	15.80%	2.16e-05
DIT	0.18	0.10	80.00%	5.41e-06
LCOM	0.16	0.15	6.70%	1.43e-03

Table 5: An average of the thresholds' values found using Shatnawi's approach.

Metric	Context threshold	Control threshold
WMC	1.79	1.82
NOC	0.75	0.74
CBO	1.81	1.85
RFC	3.79	3.90
DIT	0.92	0.93
LCOM	0.62	0.62

In Table 6, we present the average *F-measure* of the two data samples. Furthermore, we present the improvement of the threshold definition, $\theta = \Omega/\Psi$, and the *p-value* resulting from the Wilcoxon test. By analyzing the results, we can notice that for all metrics, the threshold definition was slightly improved by considering the context. This is possible because the thresholds were defined for different samples. Since for all metrics $p - value < 0.05$, we refute the null hypothesis that states that considering the context to define the thresholds does not influence its definition for the solution presented by Shatnawi [9].

Table 6: Results for Shatnawi's approach.

Metric	F-measure		θ	*p-value*
	Context	Control		
WMC	0.40	0.31	29.03%	5.41e-06
NOC	0.09	0.06	50.00%	5.41e-06
CBO	0.39	0.30	30.00%	5.41e-06
RFC	0.39	0.31	25.81%	5.41e-06
DIT	0.42	0.32	31.25%	5.41e-06
LCOM	0.34	0.28	21.43%	9.08e-05

5. Threats to Validity

A threat to internal validity is that we did not consider the impact of many software development factors such as development process and team experience. In addition, the experiment would strengthen the evidence by performing

the same tests alternating the control and context groups, showing the threat of the control group. Finally, we only evaluated Java-based projects.

As threat to external validity, we only used data from fifteen systems and two solutions to derive thresholds, which might not be enough to generalize the data in the context of CK metrics. Furthermore, we cannot generalize the results for other software metrics.

As threat to construct validity, we identified a faulty class through regular expressions in commit messages using the BugInfo tool, which might not be accurate. In addition, we used the percentage of faults to validate the thresholds. Even though there is evidence that the six CK metrics can indicate flaws in software [13] and is the same process used by Shatnawi [9], the reliability of the metric can be a threat to the validity of our study.

6. Conclusions

In this paper, we presented results of an empirical study performed to evaluate the influence of the context on the definition of software metrics' thresholds. The scope of our contribution is restricted to CK metrics. We executed two quasi-experiments, each one using one solution proposed in the literature to define thresholds for software metrics. We used the solutions presented in Foucault *et al.* [6] and Shatnawi [9]. The objects of study were data from fifteen open-source Java-based projects.

We used the Wilcoxon test to evaluate both quasi-experiments. For both cases, the threshold's definition accuracy improved by considering the context to compute it. Therefore, we present evidence that, in the context of CK metrics, it is relevant to consider the context to define thresholds. As a result, we conclude that thresholds can only be reused for projects with similar contexts.

In our future work, we plan to study the implications of our results in decision-making and software quality management. Furthermore, we plan to execute experiments to investigate how, for other types of software metrics, the context influences the definition of thresholds.

References

[1] N. E. Fenton and M. Neil, "Software metrics: roadmap," in *Proceedings of the Conference on the Future of Software Engineering*. ACM, 2000, pp. 357–370.

[2] M. J. Ordonez and H. M. Haddad, "The state of metrics in software industry," in *Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on*. IEEE, 2008, pp. 453–458.

[3] T. L. Alves, C. Ypma, and J. Visser, "Deriving metric thresholds from benchmark data," in *2010 IEEE International Conference on Software Maintenance*. IEEE, sep 2010, pp. 1–10.

[4] K. A. Ferreira, M. A. Bigonha, R. S. Bigonha, L. F. Mendes, and H. C. Almeida, "Identifying thresholds for object-oriented software metrics," *Journal of Systems and Software*, no. 2, pp. 244–257, feb 2012.

[5] R. Martin, "Oo design quality metrics," *An analysis of dependencies*, vol. 12, pp. 151–170, 1994.

[6] M. Foucault, M. Palyart, J.-R. Falleri, and X. Blanc, "Computing contextual metric thresholds," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing - SAC '14*. New York, New York, USA: ACM Press, mar 2014, pp. 1120–1125.

[7] B. Efron, "Bootstrap methods: another look at the jackknife," *The annals of Statistics*, pp. 1–26, 1979.

[8] R. Shatnawi, W. Li, J. Swain, and T. Newman, "Finding software metrics threshold values using ROC curves," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 22, no. 1, pp. 1–16, jan 2010.

[9] R. Shatnawi, "Deriving metrics thresholds using log transformation," *Journal of Software: Evolution and Process*, vol. 27, no. 2, pp. 95–113, feb 2015.

[10] F. Zhang, A. Mockus, Y. Zou, F. Khomh, and A. E. Hassan, "How Does Context Affect the Distribution of Software Maintainability Metrics?" in *2013 IEEE International Conference on Software Maintenance*. IEEE, sep 2013, pp. 350–359.

[11] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *Software Engineering, IEEE Transactions on*, vol. 20, no. 6, pp. 476–493, 1994.

[12] Z. Bukhari, J. Yahaya, and A. Deraman, "Software metric selection methods: A review," in *Electrical Engineering and Informatics (ICEEI), 2015 International Conference on*. IEEE, 2015, pp. 433–438.

[13] 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*. ACM, 2010, p. 9.

[14] P. Oliveira, F. P. Lima, M. T. Valente, and A. Serebrenik, "Rttool: A tool for extracting relative thresholds for source code metrics," in *2014 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2014, pp. 629–632.

[15] P. Oliveira, M. T. Valente, and F. P. Lima, "Extracting relative thresholds for source code metrics," in *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*. IEEE, feb 2014, pp. 254–263.

[16] L. Sánchez-González, F. García, F. Ruiz, and J. Mendling, "A study of the effectiveness of two threshold definition techniques," in *Evaluation & Assessment in Software Engineering (EASE 2012), 16th International Conference on*. IET, 2012, pp. 197–205.

[17] R. Bender, "Quantitative risk assessment in epidemiological studies investigating threshold effects," *Biometrical Journal*, vol. 41, no. 3, pp. 305–319, 1999.

[18] S. K. Thompson, "Simple random sampling," *Sampling, Third Edition*, pp. 9–37, 2012.

[19] M. Jureczko, "Significance of different software metrics in defect prediction," *Software Engineering: An International Journal*, vol. 1, no. 1, pp. 86–95, 2011.

[20] L. Madeyski and M. Jureczko, "Which process metrics can significantly improve defect prediction models? an empirical study," *Software Quality Journal*, vol. 23, no. 3, pp. 393–422, 2015.

[21] D. F. Bauer, "Constructing confidence sets using rank statistics," *Journal of the American Statistical Association*, vol. 67, no. 339, pp. 687–690, 1972.

Who Will be Interested in? A Contributor Recommendation Approach for Open Source Projects

Xunhui Zhang, Tao Wang, Gang Yin, Cheng Yang, and Huaimin Wang
College of Computer Science
National University of Defense Technology
Changsha, Hunan, China
{zhangxunhui,taowang2005,yingang,hmwang}@nudt.edu.cn, delpiero710@126.com

***Abstract*—The crowds' continuous participation and contribution are the key factors for the success of open source projects. However, among the massive competitors, it is difficult for a project to attract enough contributors by just passively waiting for enthusiasts to join in. Instead, it should actively seek gifted developers. Most of the current studies mainly focus on recommending experts inside a repository for some specific development tasks. In this paper, we propose a novel approach *ConRec* to recommend potential contributors across the entire open source community for given projects. It leverages the developers' historical activities in projects to analyze their technical interests and technical connections with others. Thereafter, it combines collaborative filtering algorithm with text matching algorithm to recommend proper developers. We conducted extensive experiments on 5,995 open source projects and 2,938,620 developers in GitHub. The results show that the proposed algorithm can recommend contributors to open source projects with the best performance of 63% in accuracy, and solve the cold start problem as well.**

Keywords-Contributor Recommendation; Collaborative Filtering; Text Matching; GitHub

A. Introduction

Open source software (OSS) has become increasingly popular in software development. Quite different from the traditional software development, OSS is driven by massive crowds including developers, users, managers and so on. These stakeholders involve in OSS by interests, and most of them have their own full-time job and can only spend spare time on OSS. They can join in or withdraw from it at any time. Nevertheless, OSS has achieved great success at creating high-quality software like Linux, MySQL, Spark and so on, and is viewed as "eating the software world" by "the Future of Open Source Survey" [1].

The behavior of open source itself does not necessarily result in a project's success, but the continuous participation and active contribution from the crowds do play a crucial role. In GitHub, there are more than 48 million open source projects. However, over 2 million projects are not forked or watched by anyone after releasing, and 15.1% of them were not updated for more than a year and finally failed. Even for those projects which used to be successful will languish without continuous contribution of developers.

Therefore, finding and attracting the right developers to participate is quite crucial for OSS. On the one hand, proper developers can provide necessary technical expertise that a project needs; On the other hand, suitable projects can inspire developers to participate in and to contribute their innovations continuously. However, there is a massive amount of competitive OSS, and developers are often limited by their time and energy to browse and choose from all the related projects. An automatic approach to bridge the gap between developers and projects and match them means much for both developers and projects.

Several studies have explored the act of recommending developers to open source projects. Nguyen et al. [2] and Ma et al. [3] proposed approaches to fix the recommendation issues by analyzing the implementation history and expertise of developers. Thongtanunam et al. [4] and Yu et al. [5] focused on the automatic pull-request reviewer recommendation problem by analyzing the development history and social connections of individuals in the software community. Most of these studies mainly focus on specific software tasks and limit the recommended candidates to the core developers of the projects. By contrast, the current study is rooted in the granularity of open source repositories, and our output allows recommendations of external experts throughout the community.

This study developed a general algorithm called *ConRec* for recommending suitable contributors to open source projects based on a collaborative filtering algorithm and text matching based method. We performed extensive experiments to compare the performances of different algorithms on various types of projects. The test data included approximately six thousand projects and over three million candidates in GitHub. *ConRec* performed well in all types of projects while solving the "cold start" problem. The main contributions of this study are as follows:

- We design a commit network to measure the collaboration connections between developers. Based on this network,

DOI reference number: 10.18293/SEKE2017-067

we develop a weighted collaborative filtering (*WCF*) algorithm to recommend.
- We design a text matching based recommendation algorithm based on the text information of projects. It can solve the cold start problem effectively for projects which have only few pre-existing developers before recommendation.
- We combine the above two algorithms together and design a hybrid approach called *ConRec*. We conduct extensive experiments on over 5,990 projects and about three million developers in GitHub, and achieves recommendation accuracy of about 63%.

The rest of this paper is organized as follows. Section 2 reviews a few related studies. Section 3 describes the framework of *ConRec* and the detailed recommendation algorithms. Section 4 presents the design of the experiments. Section 5 discusses the experiment results. Section 6 elaborates the conclusion and describe the future plans of the study.

I. Related Work

A. Expert Identification and Prediction

In software development, the expertise of developers is one of the most important factors that influence the efficiency and quality of software development. Hence, an accurate expert identification is of immense importance in global and distributed software development. Many studies have explored expert identification and prediction from different perspectives. In references [2][3], Nguyen and Ma, et al. evaluated the expertise of developers by analyzing the usage frequency of methods and time distribution on issue fixing, respectively. Thereafter, they designed the corresponding metrics to measure the technical capabilities of developers from the aspects of usage and implementation. Gharehyazie et al. [6] leveraged the activities of participants in a mailing-list communication network and issue fixing history, as well as proposed an approach to predict the probability of becoming a developer in a project. Robbes [7] used the interactions between developers to calculate their expertise. Schuler and Zimmermann [8] measured the expertise of developers by using the frequency of usage of specific application programming interface.

Most of these studies mainly focused on the capability of developers to solve problems, thereby failing to consider the probability of participating in target projects. The current study combines the ability of developers and the association among them to find suitable committers to projects.

B. Task Assignment

The development of social coding communities has resulted in collaboration and cooperation becoming particularly common. Moreover, the rapidly increasing numbers of repositories and tasks pose difficulties for developers.

Automatic task assignment has been extensively studied. In particular, bug assignment has attracted significant research interest. Bhattacharya [9], [10], Naguib [11], Xia [12], Shokripour [13], and others introduced many methods to recommend experts to solve bugs. Yu et al. [5], [14], [15] analyzed the social coding pull-request mechanism and developed methods for pull-request recommendation. Balachandran and Thongtanunam et al. [16][4] generated methods for code reviewing in social coding communities. Kagdi [17] heuristically created a code reviewer recommendation system using the history of activities and expertise of developers.

These studies mainly focused on the fine-grained tasks of open source repositories. By contrast, the present research is concerned with finding suitable contributors for a project, which is a coarse-grained topic.

C. Collaborative Network in Social Coding Communities

Although social coding is quite different from traditional software development, the working habits of developers are quite similar. People tend to work in groups and form new groups with familiar cooperators, thereby leading to the formation of a collaborative network.

Hertel et al. [18] studied the motivation of participants in OSS, and determined that people tend to work as a team. Grewal et al. [19] explored the network embeddedness of open source projects and realized that strong embeddedness leads to success. Weiss et al. [20] revealed that developers will migrate from one community to another with other collaborators. Hahn et al. [21] used these studies as basis to indicate that prior social relations in a network of developers can assist to attract additional developers. They also determined that developers tend to join in projects with which they have strong collaborative ties [22].

In considering the existing studies, we learned that developers in OSS tend to form collaborative networks. Furthermore, cooperators tend to form groups when developing new projects. On the basis of this phenomenon, we developed the following recommendation algorithm.

II. Method

In this section, we present the framework of *ConRec* and introduce the detailed mechanism of the proposed algorithm.

A. Intuition of ConRec

The basic idea stems from the observation of the relationship between developers and projects. Hahn [22] and Madey et al. [23] determined that developers in social coding communities tend to form groups, and that these collaborative networks affect the choice of participation of developers. We extract a part of the commit network in Figure 1.

Figure 1 shows that developers who have committed to the same project tend to collaborate in other projects. The same is true real life; when developing a project, developers form groups, become familiar with one another, and gain mutual trust. Therefore, they are likely to collaborate in other projects. Figure 1 also shows that developers who focus on "cocos2d" tend to work on the same projects. This condition indicates that similar technical interests lead to similar behaviors of participation. The same is true in reality, in which developers with

Pre-existing developer: developer who has already committed to the project.

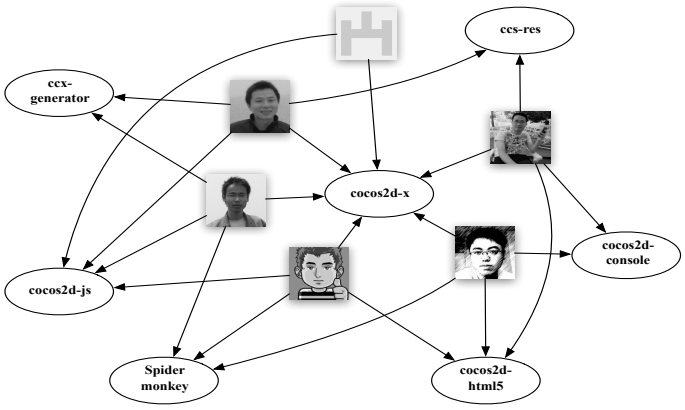


Fig. 1: Part of Commit Network

different skills or in different research fields focus on different areas. Accordingly, we developed the *ConRec* algorithm for recommending suitable contributors to open source projects.

B. Framework of ConRec

ConRec involves three steps: gathering prepared information for the recommendation algorithm, calculating relations between potential developers and target projects, and ranking and obtaining the final results. Figure 2 illustrates the framework.

1) Gather Prepared Information: The first step is to gather prepared information for the target project, including its pre-existing developers, potential developers, major programming language, and the technical terms in the project name and description.

2) Calculate Relation: The second step is to perform the recommendation task for the target project based on the prepared information. We use the *WCF* algorithm and text matching based algorithm to calculate the relation between potential developers and projects. Thereafter, we consolidate the results by merging the recommendation results of *WCF* with the text matching results.

3) Rank Results: The third step is to obtain the final recommendation results of the target project. After calculating the relation of developers, we rank the results in descending order and identify the top-k developers as the final result.

C. WCF Algorithm

This part describes the *WCF* algorithm in detail, which comprises two steps. Figure 3 shows the work flow of this algorithm.

1) Expert Selector: The first step is to select the developers who are familiar with the major programming language of the target project. If the developer lacks experience in the target programming language, then he/she may lack interest and experience difficulty committing to the target project. Therefore, we count the number of times a developer commits with the target programming language, and we consider that potential developers should commit at least four times to projects which use the target language as the major programming language. The reason for opting for this value as the threshold is discussed in research question 3.

2) WCF: The second step is to recommend developers based on the collaborative filtering algorithm. Here, we select the pre-existing developers from the prepared information gathered in the first step of *ConRec*. Thereafter, we calculate the relation between each pre-existing developer and potential developer based on their relation to other projects. Equation 1 shows the relation between a developer and a project where the first and second parameters denote the developer and project, respectively. $Commit(d,p)$ refers to the number of times developer d commits to project p, and U_p denotes the set of pre-existing developers for project p.

$$R_{dp}(d,p) = \frac{Commit(d,p)}{\sum_{i=1}^{|U_p|} Commit(U_p[i],p)} \tag{1}$$

For the developer relation, we use the Vector Space Similarity algorithm (Equation 2), where P_A refers to the projects that A have committed to.

$$R_{dd}(A,B) = \frac{\sum_{p:\{P_A \cap P_B\}} R_{dp}(A,p) * R_{dp}(B,p)}{\sqrt{\sum_{p:P_A} R_{dp}^2(A,p) * \sum_{p:P_B} R_{dp}^2(B,p)}} \tag{2}$$

After calculating the relation between the developers, we compute the relation between the developer and project according to Equation 3, where D_p denotes the pre-existing developers for project p.

$$result(A,p) = \sum_{d:D_p} R_{dp}(d,p) * R_{dd}(A,d) \tag{3}$$

Lastly, we rank the results of all potential developers for the target project in descending order, and select the top-k values as the final result.

D. Text Matching Based Recommendation Algorithm

This part aims to improve the recommendation result of *WCF* algorithm. When lacking pre-existing developers, *WCF* is unsuitable for recommendation. To solve the cold start problem, we extract the technical terms and match the developers that focus on the target techniques. The idea is based on reference [24], which uses text information to measure expertise. Figure 4 shows the work flow.

1) Generate Technical Terms: First, we generate the technical terms from the name and description of the target project using smart IKAnalyzer, and remove the words that are included in the stop-word dictionary.

2) Generate Term Map: Second, we identify the potential developers based on the technical terms of the target project. Developers who have participated in projects can get their related terms. Thereafter, we can calculate the relation between terms and developers based on the *TF-IDF* algorithm [25]. The equation is shown as below, where P_t represents the set of projects with term t, and T refers to the entire set of terms.

$$R_{dt}(d,t) = \sum_{p:P_t} R_{dp}(d,p) * log\frac{|\bigcup_{x:T} P_x|}{|P_t|} \tag{4}$$

Latent developer: developer that may commit to the project in the future.

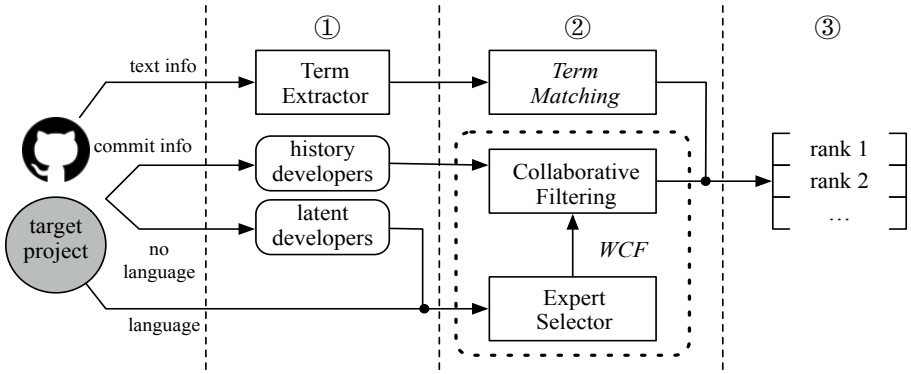


Fig. 2: Framework of *ConRec*

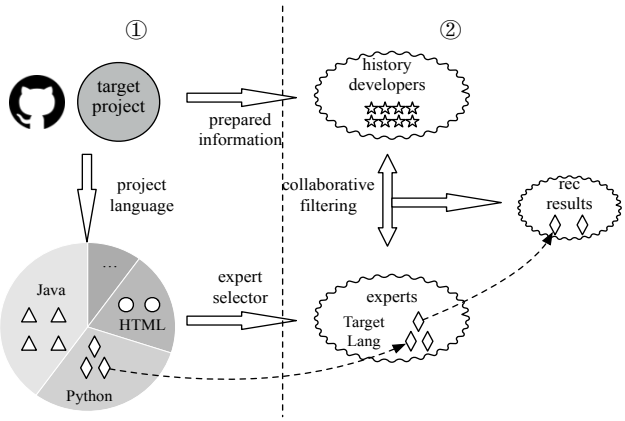


Fig. 3: Weighted Collaborative Filtering Algorithm

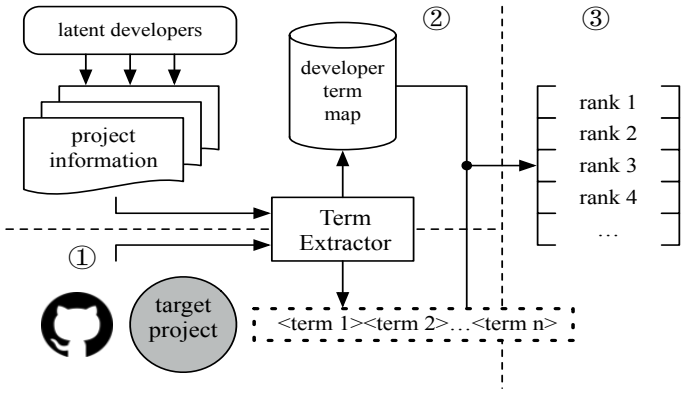


Fig. 4: Work Flow of Text Matching Algorithm

3) Rank Result: After calculating the relation between the potential developers and terms, we compute the final results using Equation 5, where T_{dp} denotes the set of terms that match developer d with project p. Thereafter, we rank the results in decreasing order. When *WCF* cannot recommend a sufficient number of developers, the text matching based algorithm serves as the supplement of the *WCF* algorithm, the recommendation result of which is used for those projects that lack of pre-existing developers.

$$result(d,p) = |T_{dp}| * \sum_{t:T_{dp}} R_{dt}(d,t) \tag{5}$$

III. Experiment

In this section, we will propose some research questions about our recommendation algorithm, and describe the experiment data set. Meanwhile, we describe the metrics that is used to validate the proposed algorithm.

A. Research Questions

We derive the following research questions to explore the relation between developers and projects based on the commit number and analyze the performance of our recommendation algorithm.

- Q1: How does *ConRec* perform compared with the two-value traditional collaborative filtering method?
- Q2: How do the algorithms differ for projects with different numbers of pre-existing developers?
- Q3: How will the number of commits affects the recommendation performance when defining the experts in given programming languages?

For Q1, we conduct experiments and compare the accuracy of different algorithms when recommending to a number of projects. For Q2, we divide the test projects into two parts. One part involves projects that have few pre-existing developers and the other involves projects with adequate pre-existing developers. Thereafter, we compare their performance. For Q3, we conduct experiments to observe the performance of *ConRec* when providing different threshold values to the expert selector part.

B. Data Set

Prior to the experiment, we generate data from the GHTorrent MySQL dump, which was released in March 2016. We firstly determine a time point. All the information prior to the time point is referred to as pre-existing information and the remaining is regarded as potential information.

http://ghtorrent.org/downloads.html
Time point: 2014-09-14.

In order to validate our recommendation algorithm, we should select well developed open source projects and determine whether our algorithm can recommend new committers to them after the time point. We select 492,590 projects, the commit numbers of which rank in the top 5% of all projects with over one committer. We consider them well-developed projects after the time point. Meanwhile, we select 450,170 projects, the number of new committers of which rank in the top 5% after the time point. Thereafter, we calculate the intersection of the projects with the top 5% commits and the projects with top 5% number of new committers after the time point. We also remove those projects created after the time point or had already been deleted or forked from others, and eventually obtain 5,995 projects. In the entire set, 888 projects have less than two committers prior to the time point; these projects are treated as ones with few pre-existing developers. The rest of the projects are deemed to have adequate pre-existing developers.

For potential developers, we remove those who have no commit experience prior to the time point; thus, from 10,132,629 users in GitHub, we arrive at 2,938,620 developers.

For the *CF* and *WCF* algorithm, we run them on the entire set of 10,132,629 developers and 28,118,416 projects.

The data set that we obtained is shown in Table I.

TABLE I: Dataset for Test

Items		Number
$Project_{CF}$ [a]		28,118,416
$Project_{test}$ [b]	$num(p.d.)$[c]≤ 1	888
	$num(p.d.) \geq 2$	5107
$Developer_{CF}$ [d]		10,132,629
$Developer_{potential}$ [e]		2,938,620

[a] $Project_{CF}$: projects used in *CF* algorithm.
[b] $Project_{test}$: projects used to test the performance.
[c] $num(p.d.)$: the number of pre-existing developers.
[d] $Developer_{CF}$: developers used in *CF* algorithm.
[e] $Developer_{potential}$: potential developers with over one commit prior to the time point.

C. Experiment Metrics

For the validation of our experiment, precision, recall, and MRR are unsuitable to evaluate the performances. For precision, we cannot ensure that the false positives (i.e., recommended developers who have not committed to the project) will not commit to the target project sometime in the future. For recall, the developers we recommend may commit to the target project later even though they have yet to commit. For MRR, committers whose commit numbers rank low are temporary, but they may surpass those leaders in the future.

We can use the accuracy value to test the effectiveness of the algorithm. If our algorithm can find a developer who commits to the target project after the time point, then we can say that the system takes effect. The accuracy value is positively correlated with the effectiveness of the algorithm. The equation of accuracy is shown below, where $hit@k$ refers to the number of hit projects when recommending k developers and $|test|$ represents the number of all the test projects.

$$accuracy = \frac{hit@k}{|test|} \tag{6}$$

New committer: developer who commit to the project for the first time.

IV. Results

In order to answer the three research questions mentioned above, we carried out three sets of experiments.

A. Overall Performance of ConRec

To answer Q1, we compare the accuracy value of our recommendation algorithm with the traditional collaborative filtering (*CF*) algorithm using the whole set of testing projects. For the *CF* algorithm, it first constructs a two-value matrix to record the commits of potential developers. Accordingly, 1 stands for the developer who has already committed to the project and 0 stands for the opposite. Thereafter, it calculates the relationship between the potential developers and the target project, and eventually ranks the results in descending order.

Figure 5 shows the result.

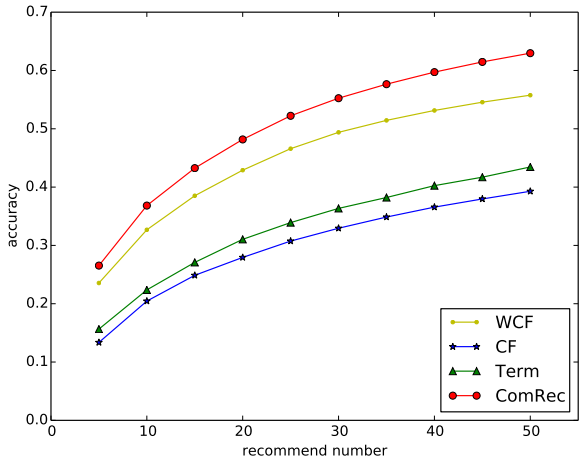


Fig. 5: Accuracy Comparison with the Entire Set of Projects

Our recommendation algorithm *ConRec* performs considerably better than the *CF* algorithm, with the accuracy of the former ranging from 26.5% to 63%. Hence, *ConRec* can meet the requirements of many projects and recommend suitable committers to them. For *CF*, the result shows that the two-value relation between committers and projects is inaccurate, that is, the algorithm cannot measure the strength of relationships.

B. Performance under Different Numbers of Pre-existing Developers

For Q2, we perform experiments and compare the performance of different algorithms in projects that have few or adequate pre-existing developers to determine whether *ConRec* can solve the cold start problem. Precisely 888 projects have few pre-existing developers. Figures 6 and 7 show the result.

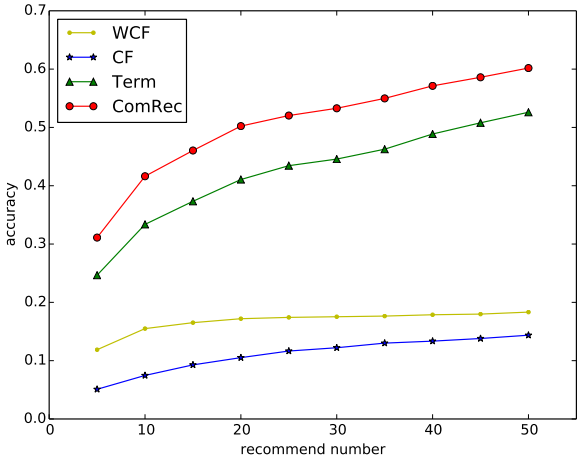


Fig. 6: Accuracy for Projects with Few Pre-existing Developers

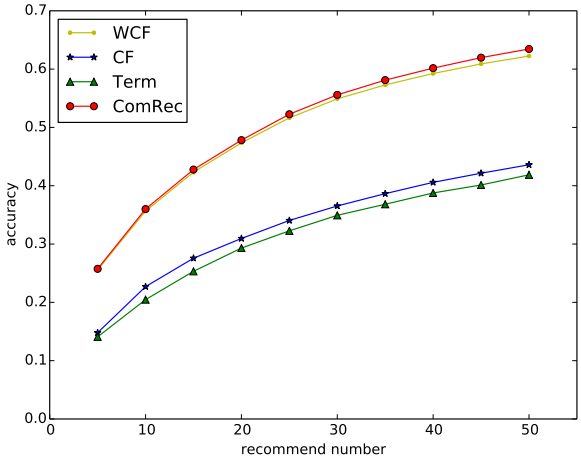


Fig. 7: Accuracy for Projects with Adequate Pre-existing Developers

Figure 6 shows that the *CF* and *WCF* algorithms do not perform satisfactorily when recommending developers to projects that have few pre-existing developers. Hence, collaborative filtering is not suitable for newly joined or unpopular projects. The text matching based algorithm shows a good performance, but the performance of *ConRec* is superior. For projects with adequate pre-existing developers (Figure 7), *ConRec* achieves the best performance, followed by *WCF* and *CF*. *ConRec* evidently performs better than *WCF*; thus, the complement of the text matching based results can facilitate the improvement of the recommendation results even for projects with adequate pre-existing developers. The *CF* algorithm performs better than the text matching based algorithm, thereby indicating that a collaborative network is more suitable than text information when recommending. This factor is our basis for using the text matching based algorithm to complete the matching result of *WCF*. The text matching based algorithm performs better than *CF* when recommending developers to an entire project set (see Figure 5). The reason is that, the number of projects that have few pre-existing developers is 888, which comprises 14.8% of all the test projects.

Therefore, the terms generated from the text information of the projects can facilitate solving the cold start problem and improve the recommendation results for projects that have few or adequate pre-existing developers.

C. Performance of ConRec in Different Threshold Values of Expert Selector

Considering about Q3, we perform a contrast experiment with different threshold values of the expert selector. Figure 8 shows the accuracy of *ConRec* when recommending 50 developers, where the x- and y-axes represent the threshold and accuracy values, respectively. We set the threshold value range from 2 to 30 with every other number.

Figure 8 shows that when the threshold value is set to 4, the accuracy performs the best. Hence, when a developer commits

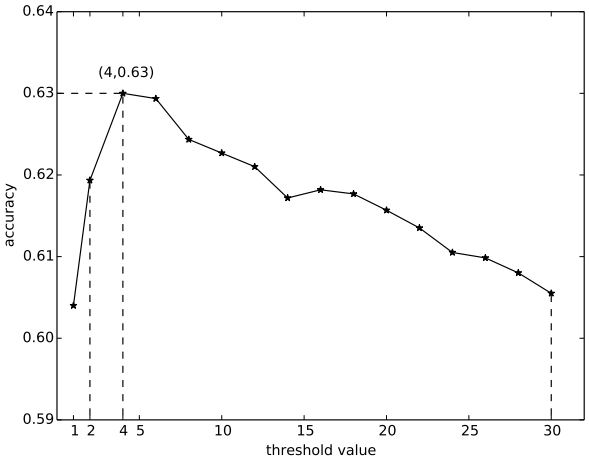


Fig. 8: Accuracy Change with Different Threshold Values

over 4 times with the target language, we can regard him/her as an expert. For threshold values below 4, the accuracy is not that good because developers who are unfamiliar with the programming language remain included. For threshold values over 4, the performance decreases with the increase of the threshold value. The reason is that *ConRec* filters accurate developers and use text matching based algorithm to complete the recommendation result.

In conclusion, the threshold value of the expert selection part influences the performance of *ConRec*. If the value is considerably small, then developers who are unfamiliar with the target programming language are not filtered. If the value is substantially large, then a few related developers are filtered, and the text matching based algorithm is initiated. However, the results are not as good as those of the collaborative filtering algorithm. Therefore, we select the threshold value to 4 in

ConRec.

V. Conclusion and Future Work

The study aims to develop a general algorithm for recommending suitable committers to open source projects based on a collaborative network. We derive three research questions and perform many experiments on 5,995 popular projects in GitHub. The results indicate the superior performance of *ConRec*. *ConRec* somewhat solves the cold start problem by combining the collaborative filtering and text matching algorithms. *ConRec* is suitable for different open source communities because it simply considers the commit information and the text information of a project, including the name, description and language. These details are common information in social coding communities.

However, it still has a few limitations.

Expert selection is relatively simple, that is, the programming skill of a developer is more related to the quantity of code than to the number of commits.

Many projects involve over one programming language. Therefore, developers may commit to a project without using the major programming language.

Many short-term developers engage in open source projects; hence, they are likely to commit to a project in a short time. Therefore, the algorithm should also consider the commit time of pre-existing developers. For example, commits from a long time ago may not be considered.

Different types of activities of developers like fork, watch and star may also be considered. Different types of developers can simultaneously participate in different numbers of projects.

For our future study, we will firstly build up a prototype system by implementing our algorithm. Thereafter, we improve the algorithm by simultaneously considering these limitations and iterating the system.

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.61502512,61432020,61472430,61532004).

References

[1] M. Silic, "Dual-use open source security software in organizations–dilemma: Help or hinder?" *Computers & Security*, vol. 39, pp. 386–395, 2013.

[2] T. T. Nguyen, T. N. Nguyen, E. Duesterwald, T. Klinger, and P. Santhanam, "Inferring developer expertise through defect analysis," in *Software Engineering (ICSE), 2012 34th International Conference on*. IEEE, 2012, pp. 1297–1300.

[3] D. Ma, D. Schuler, T. Zimmermann, and J. Sillito, "Expert recommendation with usage expertise," in *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*. IEEE, 2009, pp. 535–538.

[4] P. Thongtanunam, C. Tantithamthavorn, R. G. Kula, N. Yoshida, H. Iida, and K.-i. Matsumoto, "Who should review my code? a file location-based code-reviewer recommendation approach for modern code review," in *Software Analysis, Evolution and Reengineering (SANER), 2015 IEEE 22nd International Conference on*. IEEE, 2015, pp. 141–150.

[5] 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.

[6] M. Gharehyazie, D. Posnett, and V. Filkov, "Social activities rival patch submission for prediction of developer initiation in oss projects," in *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*. IEEE, 2013, pp. 340–349.

[7] R. Robbes and D. Röthlisberger, "Using developer interaction data to compare expertise metrics," in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 297–300.

[8] D. Schuler and T. Zimmermann, "Mining usage expertise from version archives," in *Proceedings of the 2008 international working conference on Mining software repositories*. ACM, 2008, pp. 121–124.

[9] P. Bhattacharya and I. Neamtiu, "Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging," in *Software Maintenance (ICSM), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1–10.

[10] P. Bhattacharya, I. Neamtiu, and C. R. Shelton, "Automated, highly-accurate, bug assignment using machine learning and tossing graphs," *Journal of Systems and Software*, vol. 85, no. 10, pp. 2275–2292, 2012.

[11] H. Naguib, N. Narayan, B. Brügge, and D. Helal, "Bug report assignee recommendation using activity profiles," in *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*. IEEE, 2013, pp. 22–30.

[12] X. Xia, D. Lo, X. Wang, and B. Zhou, "Accurate developer recommendation for bug resolution," in *Reverse engineering (WCRE), 2013 20th working conference on*. IEEE, 2013, pp. 72–81.

[13] R. Shokripour, J. Anvik, Z. M. Kasirun, and S. Zamani, "Why so complicated? simple term filtering and weighting for location-based bug report assignment recommendation," in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 2–11.

[14] Y. Yu, H. Wang, G. Yin, and C. Ling, "Reviewer recommender of pull requests in GitHub," in *ICSME*. IEEE, 2014, pp. 609–612.

[15] 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.

[16] V. Balachandran, "Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation," in *Software Engineering (ICSE), 2013 35th International Conference on*. IEEE, 2013, pp. 931–940.

[17] H. Kagdi, M. Hammad, and J. I. Maletic, "Who can help me with this source code change?" in *Software Maintenance, 2008. ICSM 2008. IEEE International Conference on*. IEEE, 2008, pp. 157–166.

[18] G. Hertel, S. Niedner, and S. Herrmann, "Motivation of software developers in open source projects: an internet-based survey of contributors to the linux kernel," *Research policy*, vol. 32, no. 7, pp. 1159–1177, 2003.

[19] R. Grewal, G. L. Lilien, and G. Mallapragada, "Location, location, location: How network embeddedness affects project success in open source systems," *Management Science*, vol. 52, no. 7, pp. 1043–1056, 2006.

[20] M. Weiss, G. Moroiu, and P. Zhao, "Evolution of open source communities," in *IFIP International Conference on Open Source Systems*. Springer, 2006, pp. 21–32.

[21] J. Hahn, J. Y. Moon, and C. Zhang, "Impact of social ties on open source project team formation," in *IFIP International Conference on Open Source Systems*. Springer, 2006, pp. 307–317.

[22] J. Hahn and Moon, "Emergence of new project teams from open source software developer networks: Impact of prior collaboration ties," *Information Systems Research*, vol. 19, no. 3, pp. 369–391, 2008.

[23] G. Madey, V. Freeh, and R. Tynan, "The open source software development phenomenon: An analysis based on social network theory," *AMCIS 2002 Proceedings*, p. 247, 2002.

[24] R. Venkataramani, A. Gupta, A. Asadullah, B. Muddu, and V. Bhat, "Discovery of technical expertise from open source code repositories," in *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 2013, pp. 97–98.

[25] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, 2003.

A GQM-based Approach for Software Process Patterns Recommendation

Zhangyuan Meng†, Cheng Zhang†, Beijun Shen†,Wei Yin‡
†School of Software, Shanghai Jiao Tong University, Shanghai, China
‡China Aeronautical Radio Electronics Research Institute, Shanghai, China
Email: {602389789, jonathenzc, bjshen}@sjtu.edu.cn, yin_wei@careri.com

***Abstract*—A good software process can help project manager manage software development effectively and control development risks. For this reason, theory and experts' experience are concluded and put into process patterns. But it still requires human skills to search for appropriate process patterns in practice. To tackle this challenge, this paper proposes a Goal-Question-Metric (GQM) based approach to recommending software process patterns. The essential idea of this approach is to use a GQM method to design scenario questions for software process patterns, elicit the requirement of new project by answering these questions, and then recommend the optimal matching patterns to the project. In particular, we use a Latent Dirichlet Allocation model on the scenario descriptions of software process patterns to achieve a text-topic distribution, and then apply the K-means method to do text clustering, which facilitate scenario questions design a lot. We evaluate the performance of our topic clustering method by comparing it with that of the statistics method based on TF-IDF. The evaluation results show that our method contributes a high F-score which is 11.6% higher than that of the traditional TF-IDF approach. Furthermore, the average precision of recommendation can reach 57%.**

***Index Terms*—Software Process Pattern Recommendation; Machine Learning; Goal-Question-Metric**

I. Introduction

Nowadays a large number of software systems are under development, meanwhile, software process plays a key role in helping software managers develop high-quality systems and control risks. For aiding software development, process experts have designed many software process patterns [1] [2] [3], including waterfall process model, incremental process model, evolution model, scrum model, test driven development and so on. These process patterns have aggregated considerable experience and knowledge, as they are extracted from many successful projects, refined and improved constantly. Literatures [4] [5] show that these patterns do provide general process solutions that can lead to further successful software projects.

Although many useful process patterns have been designed, it is not easy to find an appropriate one for a given software project. Software development is a knowledge intensive task, and many technical, social, and environment constraints and factors can influence the software development. Choosing an appropriate process pattern is not only an intelligent work, but also a manpower consumption work. The chosen process patterns may not be appropriate if these constraints are not considered. Therefore automated process pattern recommendation with high performance is in urgent need. It can put forward suggestions for the users to choose appropriate process patterns so that the software development can be significantly improved, which means the whole process can be optimized, the redundant development activities can be eliminated, and the quality and maintainability of the software under development can be enhanced.

However, it is non-trivial to recommend software process patterns automatically in practice because of two main difficulties:

1) *How to acquire and express the software process patterns and software project requirements which can be understood by computers?* Software process patterns are summed up by process experts through practical experience. There is no pattern language standard to unify formats and styles of process patterns, which makes them unstructured and difficult for computers to understand and process. There is the same problem with the software project requirements.
2) *How to recommend appropriate software process patterns according to the project requirements automatically?* Traditionally, software process patterns are recommended by human experts. However, usually there are not enough process experts, especially in small organizations. Therefore, we need an AI expert to do it in an automatic way, where the biggest trouble lies in how to compute the matching degree between the project requirements and software process patterns in a universal way.

To tackle the above challenges, we propose a *Goal-Question-Metric* (GQM) based approach to recommending software process patterns. In this approach, we design scenario questions and answers for each software process patterns according to their description. When a new project comes, these questions are answered according to its specific requirements by the project manager (PM). By calculating the correlation between project requirements and all process patterns based on real answers and expected answers, a candidate list of software process patterns is recommended.

This paper makes the following contributions:

DOI reference number: 10.18293/SEKE2017-082

1) We present a GQM-based approach of software process pattern recommendation. Through questionnaires, we extract the structured scenario data from the textual description of software process patterns and project requirements. And then, the correlations between requirements and process patterns are calculated. As a result, software process patterns with high correlations are recommended.
2) We propose a topic clustering method to facilitate design scenario questions. We use LDA [6] to build a topic model for each sentences of software process patterns descriptions. After that, we calculate the similarities between sentences based on topic distribution, and use K-means [7] algorithm to do text clustering. With these clusters, we can design the scenario questions in a cluster-level instead of single sentence level, which can save lots of time and labor.

The remainder of this paper is organized as follows: Section II presents the related work; Section III describes our approach; Section IV discusses the experiments and results; Section V concludes the paper.

II. Related Work

Pattern is a common solution to a recurring problem in a given context [8]. Birukou et al. [9] divided the problem of reusing patterns into two steps, searching for patterns and selecting patterns. The problem of searching for patterns is to find appropriate patterns to solve given problems, and the problem of selecting patterns is to choose patterns to apply from a list of patterns. In the domain of software engineering, there are many different kind of patterns. Research mainly focus on the selection and recommendation of design patterns and process model.

A. Selection and Recommendation of Design Patterns

R. Mustapha et al. [10] proposed a recommender system for design patterns by labelling each patterns with several key words, and ask user to input some key words and about their project, then patterns are recommended according to key words matching. F. Palma et al. [11] proposed a recommendation method of software development design patterns. They extracted features manually from various design patterns, and then achieved scores of these extracted features by questionnaire. The matching degree between requirements and design patterns was calculated by the sum of answered weights. Sanyawong et al. [12] extracted names of classes and methods from software design. With these names, the similarity between different software designs can be calculated, and design patterns were recommended to novice designers. Issaoui et al. [13] used semantic information of class names, method names and description of software to do recommendation.

B. Selection and Recommendation of Process Model

In recent years, with the rapid development of the Internet and open source communities, lots of historical data about the software projects has been saved which can be used by researchers. Little et al. [14] proposed several attributes to score the suitability of the development process approach for a particular project. Egwali and Akwukwuma [15] proposed 35 criteria and a relative rating mechanism for these criteria to select appropriate process model. But these criteria rely heavily on the authors' subjective opinions which may affect the selection appropriate process models. Song et al. [16] defined a machine learning based method of software process model recommendation. Choosing an appropriate process model for a new project is achieved by utilizing the relationship between software project characteristics and the appropriate process models.

In summary, most of current research on patterns selection and recommendation mainly adopt features extracted from structured data. Kubo et al. [17] proposed a method to search appropriate unstructured patterns according to patterns' popularity, but this method can't match the requirement of a software project in detail.

III. Approach

A. Approach Overview

In order to deal with above problems, we propose a GQM-based approach to software process pattern recommendation, as shown in figure 1. The input is a reusable software process pattern library, where each pattern is described in the form of *Name*, *Intent*, *Domain*, *Solution*, *Initial Context* [4], as illustrated in Table I. The output is a recommended pattern list to a new project.

TABLE I: Description Example of Pattern FLOOT.

Item Name	**Content**
Name	Full Life Cycle Object-Oriented Testing, FLOOT
Intent	FLOOT methodology is a collection of testing techniques to verify and validate object-oriented software. The goal is to define software defects before delivery to users and assure your software applicable as a complete artifact.
Domain	Testing
Solution	1. Make testing plan for systematic testing; 2. Input systematic testing plan; 3. Commit systematic testing, including functional testing, pressure testing, installation testing and operation testing; 4. Record problem and defect for regression testing. Condition 1: If it passes regression testing, go to step 5; Condition 2: If it fails to pass regression testing, go to step 1. 5. After systematic testing, software is under user testing, including Alpha/Beta testing, delivery testing.
Initial Context	1. Your software is under package for delivery. FLOOT will test software as a whole for installation tools, documents and software. 2. Master test / quality assurance plans are finished. You need to manage and track testing work.

Our approach consists of two phases:

1) At questions and answer design phase, we preprocess scenario descriptions of process pattern, build topic model for each scenario sentence. After that, we do text

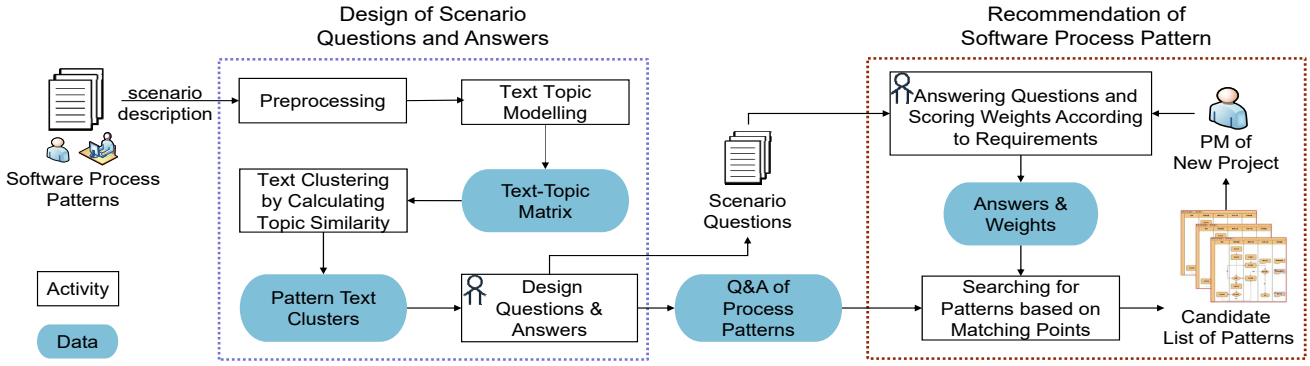


Fig. 1: Our Approach to Software Process Pattern Recommendation

clustering on all sentences for patterns by calculating similarity on text topic matrix. For each cluster, scenario questions are designed, and each question on every related pattern is assigned with an answer.

2) At pattern recommendation phase, PM firstly needs to answer related scenario questions and score the relevance between their requirements and questions. We get the candidate list of patterns recommended by calculating the matching degree between PM's answers and pattern answers designed in the previous phase.

B. Design of scenario questions and answers

Software process patterns are accumulated by experts' past experience and knowledge. However, such textual descriptions cannot be understood by machine. We adopt scenario questions to describe process pattern characteristics and new project requirements, for the further recommendation. Here, it will be described in details how the questions and answers are designed using the GQM model and machine learning technology.

1) Goal-Question-Metric Model: GQM [18] is a three layer model, where goals should be identified first, questions are designed to achieve these goals, and then questions are answered by metrics. In this paper, we apply GQM model to the questions design of process patterns. As Figure 2 shows, goals on the top layer are process patterns, questions on the middle layer are scenario questions, and metrics on the bottom layer are the answers to scenario questions, which are in the form of (Yes / No / Don't know) with a weight on a scale of 0-10. Each pattern has its scenario questions and corresponding answers.

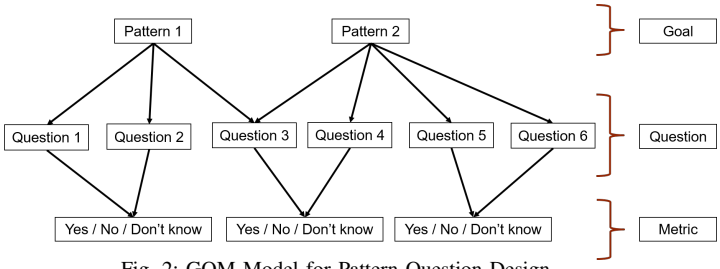


Fig. 2: GQM Model for Pattern Question Design.

2) Steps of Question Design: There are four steps in scenario question design.

- **Preprocessing**. The sentences in the *Intent*, *Domain* and *Initial Context* from pattern description (Table I) are extracted and filtered as the scenario descriptions of a pattern, and then represented in the form of bag of words.
- **Text topic modeling**. We apply LDA [6] on all the scenario sentences from all patterns to build topic model, and get text-topic probability distribution matrix. The size of matrix is d $\times$ k, where d means the amount of all sentences and k indicates the count of topics (in our method we set k with 100). Each row vector in matrix means topic probability distribution of one sentence, and the whole matrix represents topic probability distribution of all pattern descriptions.
- **Calculating topic similarity and sentences clustering**. We adopt K-means [7] to probability distribution matrix for sentences clustering. The similarity between texts is related to the similarity between vectors in matrix, which is calculated by Euclidean distance, as defined in formula (1).

$$distance(X,Y) = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \qquad (1)$$

where X,Y is the row vector which consists of x_i and y_i; x_i and y_i are respective probability of the sentence belonging to the i^{th} topic; and k is the number of all topics. After text clustering to all the pattern description sentences, we can obtain the topic cluster collection of all the sentences. In our method, we set K for K-means with 25.

- **Design of questions and answers**. We design scenario questions according to all clusters of pattern descriptions. Each pattern has one or more questions. As the example in Figure 3 shows, each pattern description consists of several sentences. After topic clustering, each cluster contains various sentences from different patterns. Questions are mainly from the clusters sentences. Once all questions have been designed for all clusters, we assign a right answer to each question on every related pattern.

3) An Example: Here is an example of how to design scenario questions and answers. Table II shows the scenario

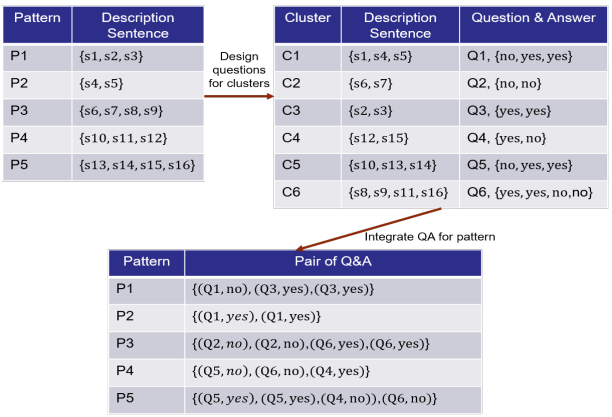


Fig. 3: Patterns, Clusters and Questions.

descriptions of three patterns, waterfall development (WD), iterative development (ID), and prototype development (PD). Table III lists the topic clustering results and their designed questions. And Table IV shows the scenario questions and answers for each pattern, where '—' means the question is not related to the pattern.

TABLE II: EXAMPLE OF SCENARIO DESCRIPTION.

Pattern Name	Pattern Description Sentences
Waterfall development	1. Be useful for complex system whose requirements should be stable and clear. 2. Be suitable for the projects with less risks. 3. Places emphasis on documentation. 4. Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.
Iterative development	5. The requirements could be not so stable at the beginning. 6. Offer many different available versions. 7. Often contact with users to obtain feedback. 8. Can reach a high demand of risk control.
Prototype development	9. Be suitable for small project whose requirements are not very clear. 10. Work with users during the development. 11. Ensure the developed software with a high usability. 12. May delay the whole development cycle.

TABLE III: TOPIC CLUSTERING AND QUESTION RAISE.

Sentence Clusters	Scenario Questions
{1,5,9}	1. Are the requirements of your project stable and clear?
{2,8}	2. Does your project have some high risks need to control?
{3}	3. Do you need detailed documents?
{4,12}	4. Do you need a complete development and strict plan?
{6}	5. Do you need to release system versions quickly?
{7,10}	6. Do you need to contact with users often during development process?
{11}	7. Does your project need a high usability?

C. Recommendation of Software Process Pattern

In this phase, appropriate patterns will be recommended according to the requirement of a new project. Our approach

TABLE IV: EXAMPLE OF Q&A DESIGN FOR PATTERNS.

Question	WD	ID	PD
1. Are the requirements of your project stable and clear?	Yes	Yes or No	No
2. Does your project have some high risks need to control?	No	Yes	—
3. Do you need detailed documents?	Yes	—	—
4. Do you need a complete and strict development plan?	Yes	—	No
5. Do you need to release system versions quickly?	—	Yes	—
6. Do you need to contact with users often during development process?	—	Yes	Yes
7. Does your project need a high usability?	—	—	Yes

obtains project requirements by questionnaire and uses scenario questions to match process patterns. The benefit of questionnaire is that it avoids analyzing the textual requirements of a new project by nature language processing technology, and helps understand them more accurately.

1) Recommendation Method: Figure 4 shows the recommendation method of software process pattern. When a new project coming, PM will answersthe scenario questions with "Yes/No/Don't know" and score the weights between the project requirements and questions. The weight is an integer from 0 to 10 for quantification of the relationship between patterns and project requirements. After recording the answers and weights, we can calculate all the matching degrees between patterns and questions. As a result, we sort all the matching degrees and pick the first five process patterns as the candidate list to PM. In addition, it is not all questions that should be answered. Questions are often correlative, so we will filter the remaining questions based on the answers of previous questions.

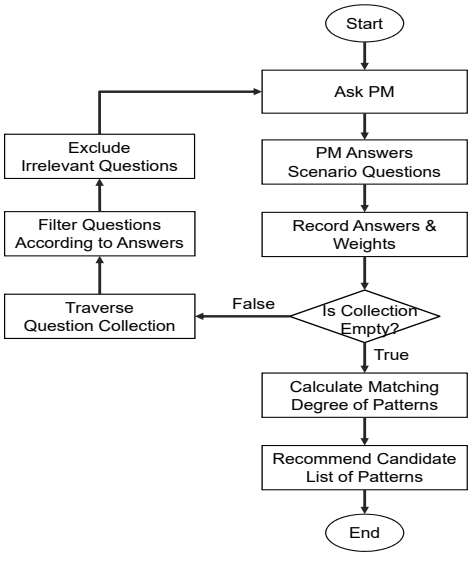


Fig. 4: Recommendation Method

To recommend the appropriate patterns for a project, the matching degree between their answers of scenario questions is calculated by formula (2).

$$MD = \frac{1}{NQ}(\sum_{i=1}^{n}(Weight_{i\_same)} - \sum_{j=1}^{m}(Weight_{j\_diff)}) \quad (2)$$

In the formula, MD is the matching degree between patterns and new project requirements; NQ is the number of all related questions for the pattern; $\sum_{i=1}^{n}(Weight_{i\_same)}$ is the weight sum of all PM's answers in accordance with pattern's answers; $\sum_{j=1}^{m}(Weight_{j\_diff)}$ is the weight sum of all PM's answers inconsistent with pattern's answers. For example, there is a question collection {Q_1,Q_2,Q_3} and their pattern answers are {yes,no,no}. A PM answers these questions and gives {(yes,10),(yes,8),(no,9)}. The questions with the same answers are Q_1 and Q_3, while Q_2 is the question with different answer. As a result, $\sum_{i=1}^{n}(Weight_{i\_same\_answer)}$ is 19 and $\sum_{j=1}^{m}(Weight_{j\_diff\_answer)}$ is 8. So the matching degree between this pattern and PM's requirements is $\frac{11}{3}$.

2) An Example: Here is a small example to illustrate which pattern is prefered. Company A needs to develop a graphic software. The requirement is not very clear and the client ask for a high usability. In order to get appropriate software process patterns, PM needs to answer the scenario questions. Table V shows the records of pattern recommendation questionnaire. In the table, column 'A' represents the answer to question and 'W' is the weight. Three patterns are examined, including waterfall development pattern (WD), iterative development pattern (ID), and prototype development pattern (PD).

Take waterfall development pattern as an example, its related question collection is {Q_1,Q_2,Q_3,Q_4} and corresponding answers are {yes,no,yes,yes}. PM's answer pairs are {(no,8),(yes,5),(no,5),(yes,3)}. Only Q_4 is matched. So Weight$_{same_answer}$ is 3, Weight$_{diff_answer}$ is 18 and the matching degree between pattern waterfall development and PM's requirements is -$\frac{15}{4}$. According to the total weight, prototype development pattern is the optimal fit among these three patterns.

TABLE V: RECORDS OF PATTERN RECOMMENDATION QUESTIONNAIRE.

Question	**A**	**W**	**WD**	**ID**	**PD**
1. Are the requirements of your project stable and clear?	No	8	-8	8	8
2. Does your project have some high risks need to control?	Yes	5	-5	5	—
3. Do you need detailed documents?	No	5	-5	—	—
4. Do you need a complete and strict development plan?	Yes	3	3	—	-3
5. Do you need to release system versions quickly?	No	8	—	-8	—
6. Do you need to contact with users often during development process?	Yes	8	—	8	8
7. Does your project need a high usability?	Yes	10	—	—	10
Total Weight	—	—	-$\frac{15}{4}$	$\frac{13}{4}$	$\frac{23}{4}$

TABLE VI: PERFORMANCE COMPARISONS OF TEXT CLUSTERING.

Clusters	Num Of Sentences	TF-IDF			LDA		
		P	R	F	P	R	F
Architecture	11	0.63	0.64	0.63	0.72	0.73	0.72
Documentation	8	0.83	0.63	0.71	0.75	0.75	0.75
Design	25	0.65	0.60	0.62	0.82	0.76	0.79
Governance	19	0.71	0.53	0.61	0.72	0.68	0.70
Testing	12	0.50	0.75	0.60	0.75	0.83	0.79

IV. EXPERIMENTS

In this section, we conduct experiments to answer these two research questions:

RQ1: Compared to traditional text clustering approaches based on statistics, does our approach reach a better performance?

RQ2: Does our approach recommend appropriate software process patterns to a new project?

A. Experimental Settings

Ambler published an online software process pattern collection*. We extract totally 378 description sentences for 89 patterns from this website. For RQ1, we use Precision, Recall and F1-Measure as the evaluation criteria. For RQ2, we use HitRate to calculate the precision of pattern recommendation, as defined in formula (3), where *hitCnt* means the number of successful recommendation and *TotalCnt* means the total number of recommendation.

$$HitRate = \frac{hitCnt}{TotalCnt} \quad (3)$$

B. Text Clustering Experiment

To answer RQ1, we conduct a comparison experiment between our approach and the statistics approach based on TF-IDF. We select 75 sentences from all these 378 description sentences and divide them into 5 topics manually, as the benchmark. At the same time, the automatic text clustering is made by two approaches, where the topic number for LDA is set with 100 and the target clusters number of K-means with 5.

The experimental result is illustrated in Table VI. It shows that our approach based on LDA has higher precision and recall compared traditional TF-IDF approach, which has better effect on text clustering of software process pattern descriptions.

C. Pattern Recommendation Experiment

To answer RQ2, we collect 30 software projects which have been closed successfully and record their corresponded process patterns manually as the benchmark. We invite 18 PMs with different professional level to involve in this experiment. For each PM, we select 5 projects randomly and offer them enough time to understand the requirements of selected projects. Then they follow our approach to obtain recommended patterns.

*http://www.ambysoft.com/processPatternsPage.html

In a recommendation, if one of the top five recommended patterns matches with the recorded patterns, we regard this recommendation as a successful one.

In our method, Totally 57 questions are designed for all these 89 patterns we extract. Table VII (HR denotes hit recommendations and TR denotes total recommendations) lists the recommendation results for different PM professional levels. It shows that the average precision of pattern recommendation can reach 57%, which can meet the managers' requirements on recommending software process patterns.

TABLE VII: RECOMMENDATION RESULTS.

Professional Level of PMs	Num of PMs	HR	TR	HitRate
Beginner	4	10	20	50%
Normal	4	11	20	55%
Medium	4	13	20	65%
Advanced	4	12	20	60%
Skilled	2	6	10	60%

D. Comparison with other methods

Considering other two state-of-the-art methods mentioned in [14] and [16]. For the first method, it only uses 4 kinds of different process model, so it is not suitable to do a direct comparison. For the second method, a huge amount of historic software project data are collected to employ this model., it is hard for us to obtain satisfied data for redoing their experiments. So we just analyze the pros and cons of different methods as shown in Table VIII. Our approach can deal with unstructured textual descriptions of software patterns, and thus is more practical and can be applied in more scenarios.

TABLE VIII: COMPARISON OF DIFFERENT APPROACHES.

Method	Pros	Cons
Our Method	accurate recommendation; expandable pattern repository; unstructured data understandable	semi-automatic questions design; Rely on the answering of the PM, not so objective
[14]	Raise criteria to measure complexity and uncertainty; Divide projects into 4 categories according to complexity and uncertainty	unstable precision; divide all process patterns into 4 category, not specific enough
[16]	quantified software development attribute; accurate recommendation	inapplicable to unstructured data(such as requirements); not convenient to add new process patterns

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a GQM based software process recommendation approach. The approach applies GQM model to design scenario questions for different patterns, with the help of natural language processing and machine learning techniques. For a new project, its requirements are elicited through answering these questions, and then the optimal matching patterns are recommended. Experimental results show our approach can recommend the proper process patterns to a specific project with a high precision.

As for future work, we will explore an automatic approach to generate scenario questions from the descriptions of software process patterns. And we also plan to do more experiments to compare our approach with others, apply our approach in practice and improve it according to the feedback.

ACKNOWLEDGEMENT

Beijun Shen is the corresponding author. This research is supported by National Natural Science Foundation of China (Grant No. 61472242) and 973 Program in China (Grant No. 2015CB352203).

REFERENCES

[1] W. W. Royce *et al.*, "Managing the development of large software systems," in *proceedings of IEEE WESCON*, vol. 26, pp. 1–9, Los Angeles, 1970.

[2] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, 1988.

[3] K. Beck, "Embracing change with extreme programming," *Computer*, vol. 32, no. 10, pp. 70–77, 1999.

[4] S. W. Ambler, *Process patterns: building large-scale systems using object technology*. Cambridge University Press, 1998.

[5] S. W. Ambler, *More process patterns: delivering large-scale systems using object technology*. Cambridge University Press, 1999.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[7] J. B. Macqueen, "On convergence of k-means and partitions with minimum average variance," *Annals of Mathematical Statistics*, vol. 36, 1965.

[8] J. M. Küster, C. Gerth, A. Förster, and G. Engels, "Detecting and resolving process model differences in the absence of a change log," in *International Conference on Business Process Management*, pp. 244–260, Springer, 2008.

[9] A. Birukou, "A survey of existing approaches for pattern search and selection," in *Proceedings of the 15th European Conference on Pattern Languages of Programs*, p. 2, ACM, 2010.

[10] Y. G. Guéhéneuc and R. Mustapha, "A simple recommender system for design patterns," *Proceedings of the 1st EuroPLoP Focus Group on Pattern Repositories*, 2007.

[11] F. Palma, H. Farzin, Y. G. Guéhéneuc, and N. Moha, "Recommendation system for design patterns in software development: An dpr overview," in *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, pp. 1–5, IEEE Press, 2012.

[12] N. Sanyawong and E. Nantajeewarawat, "Design pattern recommendation based-on a pattern usage hierarchy," in *International Computer Science and Engineering Conference (ICSEC)*, pp. 134–139, IEEE, 2014.

[13] I. Issaoui, N. Bouassida, and H. Ben-Abdallah, "A new approach for interactive design pattern recommendation," *Lecture Notes on Software Engineering*, vol. 3, no. 3, p. 173, 2015.

[14] T. Little, "Context-adaptive agility: Managing complexity and uncertainty," *IEEE Software*, vol. 22, no. 3, pp. 28–35, 2005.

[15] A. O. Egwali and V. V. N. Akwukwuma, "Security framework for software process models: Measures for establishing a choice," *Asian Journal of Information Technology*, no. 1, pp. 463–471, 2012.

[16] Q. Song, X. Zhu, G. Wang, H. Sun, H. Jiang, C. Xue, B. Xu, and W. Song, "A machine learning based software process model recommendation method," *Journal of Systems and Software*, vol. 118, pp. 85–100, 2016.

[17] A. Kubo, H. Nakayama, H. Washizaki, and Y. Fukazawa, "Patternrank: A software-pattern search system based on mutual reference importance," *15th Pattern Languages of Programming (PLoP)*, 2008.

[18] N. E. Fenton, *Software Metrics: A Rigorous and Practical Approach*. International Thomson Computer Press, 1996.

Cold-Start Developer Recommendation in Software Crowdsourcing: A Topic Sampling Approach

Yu Yang, Wenkai Mo, Beijun Shen†, Yuting Chen
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University, Shanghai, China
Email: {monkeydeking, jirachikai, bjshen, chenyt}@sjtu.edu.cn

***Abstract*—Recently, software crowdsourcing platforms, which provide paid tasks for developers, become attractive to both employers and developers. Developers expect to find tasks that match their interests and capabilities via crowdsourcing platforms, and thus recommender systems play important roles in these platforms. However, we still face several challenges when building a recommender system for a crowdsourcing platform. A major challenge is how to recommend tasks to *cold-start* developers whose task interaction data is not available. This paper presents a novel, topic sampling approach to tackling with the cold-start developer recommendation problem. First, it employs a general method for modeling developers and tasks, which solves the data heterogeneous issue across different platforms. After that, it casts the cold-start developer recommendation problem into a multi-optimization problem, and takes a topic-sampling based genetic algorithm to recommend tasks. More specifically, our approach is different from traditional solutions in that it leverages task descriptions and popularity-to-be, allowing new tasks to be recommended to cold-start developers. To evaluate the effectiveness of the proposed approach, we have conducted experiments on a large dataset crawled from three real-world software crowdsourcing platforms. Compared with other state-of-the-art recommendation solutions, the experimental results show that the proposed approach improves 75% of precision and recall on average.**

***Keywords*—Cold-Start Problem, Software Crowdsourcing, Topic Sampling, Developer Recommendation.**

I. Introduction

Software crowdsourcing has attracted great attentions from both industry and academia recently. An increasing number of software companies have turned to find online developers in software crowdsourcing platforms, such as OsChina[1] and Zhubajie[2], to accomplish various types of software development tasks, including architecture design, component development, testing and bug fixing. On a crowdsourcing platform, developers hunt for suitable tasks, and employers also search for qualified developers. Inappropriate developer-task matching may decrease the quality of the software deliverables. Thereafter, a recommender system needs to be designed and integrated into a crowdsourcing platform, helping users find the best fitting developers or tasks effectively.

For a software crowdsourcing recommender system, a major challenge is how to recommend tasks to *cold-start* developers (i.e., new developers) whose task interaction data is not available. Even worse, unlike movies or music items in traditional recommender systems, tasks in software crowdsourcing platforms may be insufficient in their data accumulations, since they usually have short lifetime period. Available developer behavior history information becomes rare, making the cold-start developer problem severe.

In this paper, we focus on solving the cold-start developer problem in software crowdsourcing recommender systems, i.e. recommending tasks to cold-start developers with little online behavior information. Traditional solutions conduct some questionnaire surveys [1] on cold-start developers and then build models based on results or simply recommend popular tasks to them [2]. However, it requires a lot of expert efforts to prepare good questionnaires, while developers may still forget to fill up them. Furthermore, these methods cannot recommend new tasks whose popularity are relative low to developers, even though these tasks are potentially popular in the future.

To tackle these problems, we propose a topic sampling based approach to recommend tasks to cold-start developers. Our approach builds the models of developers and tasks in a general way, casts the cold-start developer recommendation problem into a multi-optimization problem, and takes a topic-sampling based genetic algorithm to recommend tasks. More specifically, it leverages task descriptions and popularity-to-be, and can recommend even new tasks to cold-start developers, which makes it significantly different from previous solutions. We conducted several experiments to evaluate the proposed approach. Using three real datasets from software crowdsourcing platforms, we can conclude that our approach has a significant enhancement, 75% on both precision and recall in average, at the recommendation.

Our main contributions are summarized as follows:

1) *Task Popularity-to-be Estimation*. Instead of using existing task popularity, we propose an estimation method to get task popularity-to-be as one of key properties of task model. It adopts regression, a machine learning technique, to estimate whether a cold-start task may become popular in the future.
2) *General Resources Modeling*. We propose a general modeling method for modeling developers and tasks. The method abstracts the key properties of resources from heterogeneous data, modeling developers from

† Corresponding Author
[1]http://www.oschina.net/
[2]http://www.zbj.com/
DOI reference number: 10.18293/SEKE2017-104

their profiles and behaviors, while modeling tasks from their descriptions by natural language processing technologies. Then it transfers the features from warmed developers (also called non-cold start developers) to cold-start developers, through classification and clustering.

3) *Cold-start Developer Recommendation Algorithm*. With task model and developer model, a semi-supervised algorithm is designed to recommend tasks to cold-start developers, called *Topic Sampling-based Recommendation Algorithm*, which designs the recommendation problem as a new multi-optimization problem and employs the genetic algorithm to solve it.

II. Related Work

Recommendation System. The traditional approaches on recommendation are collaborative filtering (CF) approaches [3], such as singular value decomposition (SVD) [4], where the user gets items based on other items with similar patterns. However, an inherent prerequisite of CF is to have historical user-item interactions. Thus, CF suffers from the cold-start problem.

Some researchers proposed content-based approaches for building a recommender system. For example, Takacs et al. constructed content similarity matrix and used neighbor based approach to recommendation, where users ratings are affected by their nearest neighbors [5]. But it is difficult for content-based approaches to construct proper profiles for cold-start developers.

Cold-Start Problem. In this paper, we focus on the user cold-start problem in recommender systems. To model the preferences of cold-start users, previous work usually obtain related information by short interviews [6] [7]. During interviews, the users are asked to rate several items from carefully constructed seed sets, which are constructed based on popularity, contention, and coverage [8]. However, usually only a few people will fill out the questionnaires, and badly written questions limit the usefulness [9].

One of the classical approaches to addressing cold-start problem is popularity-based approach. Miao et al. jointly predicted the rating and popularity for cold-start items by sentinel user selection [10]. Arapakis et al. characterised the online popularity of news articles by two different metrics [11]. But popularity-based approaches are not good at personalizing recommendation.

Another is bandits-based approach. Li et al. modeled personalized recommendation of news articles as a contextual bandit problem [12], and solved it by using user-click feedback to maximize total user clicks. In [3], CF was combined with exploration-exploitation strategies for content recommendation. Instead of cold-start users, it is better at solving cold-start item problem.

Developer Recommendation. There are also some studies of recommendation technologies for software crowdsourcing platform. Zhu extracted skill and location features from the textual descriptions, and adopted *learning to rank* technology to perform recommendation [13]. Lee et al. proposed a dynamic planning algorithm to recommend tasks [14]. Yuen et al. pointed out that the past task preference and user performance could be utilized for task preference model [15]. However all of these work did not address the cold start problem.

III. Problem Definition and Approach Overview

We give a formal definition of cold-start developer recommendation problem, and then present an overview of our approach.

A. Cold-Start Developer Recommendation Problem

There are two basic resources on a software crowdsourcing platform: tasks denoted as T, and developers as U. Once an employer has requested a task, the developers can bid for, win, and then deliver the task. So we express the relationship between one developer u and one task t as the set $R = \{bidding, winning, delivering\}$. Thus, data on the software crowdsourcing platform can be expressed as a relationship triples: $(u, t, r) \in U \times T \times R$.

Definition 1: ***Developer Activity Percentile.*** We design a new metric, developer u's current activity percentile $Act\,(u)$, to measure how often the developer interact with tasks, which is defined as the number of tasks u has bid for.

$$Act\,(u) = \sum_{i \in T} 1\,\{if\ u\ bids\ i\} \tag{1}$$

Then, we calculate the average current activity percentile μ_{act} and the variance of the current popularity σ^2_{act} on the platform. In this paper, we suppose that the activity percentile is consistent with the Gaussian distribution $Act \sim N\left(\mu_{act}, \sigma^2_{act}\right)$.

Definition 2: ***Cold-Start Developer.*** When the activity percentile of developer u is less than twice the variance of the Gaussian function formed by all developers' activity percentile on the platform, developer u is regarded as a cold-start developer, that is, $Act\,(u) < Gaussian\left(\mu_{act} - 2 * \sigma^2_{act}; \mu_{act}, \sigma^2_{act}\right)$.

Definition 3: ***Cold-Start Developer Recommendation Problem.*** For the all tasks, it is given that $\forall t \in T, t = \{desc, f\}$, where $desc$ represents description of task t and f a boolean value indicating whether task is terminated. The problem is that recommender system must learn an appropriate model to recommend a suitable task set $T^{'}$ for a cold-start developer, where $\forall t^{'} \in T^{'} \left\{t^{'}.f = 0\right\}$. Besides, the task set recommended must meet $\forall t^{'} \in T^{'} \left\{i\ does\ not\ bid\ t^{'}\right\}$, that is, without being bid by i.

B. Approach Overview

To address the above cold-start problem, a novel approach is proposed, as shown in Fig. 1. It consists of two phases, the resource modeling phase which models tasks and developers from heterogeneous data in the platform, and the recommendation phase which recommends appropriate tasks for cold-start developers with topic sampling.

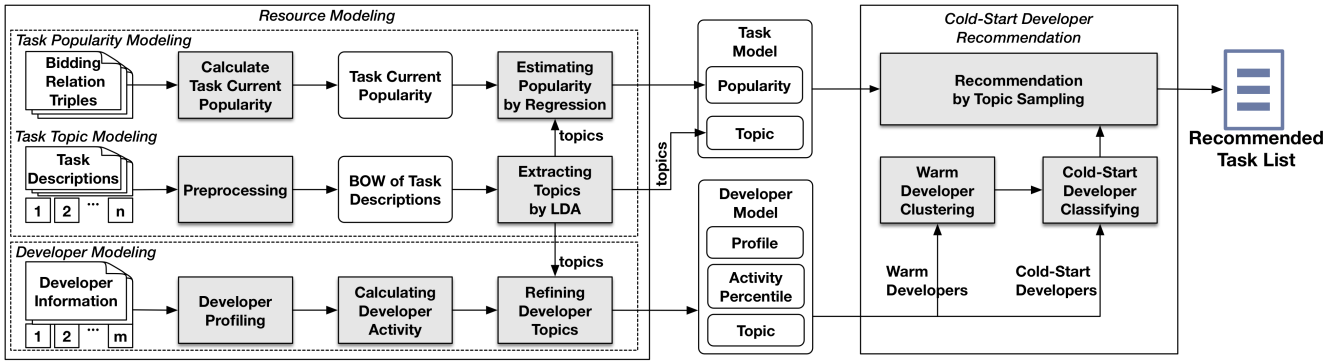


Fig. 1: Our Approach to Cold-Start Developer Recommendation.

1) Software Crowdsourcing Resource Modeling: Due to heterogeneous data from different software crowdsourcing platforms, we build the models of developers and tasks in a general way. For tasks, our approach extracts task topics by the topic model, and calculates task popularity by the regression model. For developers, it builds profile vector, topic vector and activity percentile through NLP technologies.

2) Cold-Start Developer Recommendation: Recommendation phase has three steps: (1) *Warm Developer Clustering* produces some clusters and their center topics; (2) *Cold-Start Developer Classification* assigns each cold-start developer to one cluster of warm developers; (3) *Recommendation Based on Topic Sampling* proposes a topic-based popularity sampling algorithm which applies genetic programming to perform recommendation.

IV. Software Crowdsourcing Resource Modeling

In this section, a general resource modeling method will be introduced, which can extract task and developer information from heterogeneous data in different software crowdsourcing platforms.

A. Task Modeling

One of the basic elements in software crowdsourcing is the task. Each task has its own task description including a title and a body. Although there are other properties to describe tasks in some specific platforms like tags, etc., without loss of generality, we only set task descriptions as the input of modeling phase.

1) Task Topic Modeling: Task descriptions are written in natural language by the employer. We use a popular topic modeling algorithm, Latent Dirichlet Allocation (LDA) [16], to obtain the topic distribution to represent tasks descriptions. Before applying LDA, the descriptions need to be preprocessed. We first concatenate the title and the body of a task description to a big text, and remove stop words and words whose part-of-speech tags are not noun, adjective, or verb. Then each text is represented by a bag-of-words (BOW) vector. We set the number of topics to 150, and thus, each description is finally transformed into a 150-dimensional topic vector. Distinctly our method can transform heterogeneous task data from different platforms into an isomorphic vector.

2) Task Popularity Modeling: In traditional popularity-based recommendation systems, an item's current popularity Pop_c is defined as the average rating of all users to item or how many people rate/bid it. However, it cannot reflect the popularity well especially for new tasks. A new task may be popular in the future, but at the beginning, it cannot attract enough developers in short time, and thus its current popularity is low. To better model tasks, we propose a new concept called *popularity-to-be* Pop_e, which is calculated by a regression model and extremely beneficial for new tasks.

Given the training dataset $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}, x \in R^k, y \in R$, where y_i is current popularity of a task which has been bid and delivered, and x_i the corresponding task topic vector. Regression algorithms can fit a model $\hat{f}(x) \rightarrow R$ from these historical data. Then using this model, for a task, its popularity-to-be can be estimated by the task topic vector.

There are many regression algorithms, such as Classification and Regression Tree (CART) [17], linear regression [18], Gaussian mixture model [19], etc. In the experiment, the classical CART algorithm achieves the best performance and thus we apply it to estimate task's popularity-to-be.

It is meaningful and effective to extract information from the task description for estimating popularity because the task popularity-to-be can indirectly reflect the distribution of topic. However, the longer tasks are released, the less credible the estimated popularity will be. In order to avoid the inaccuracy issue, we propose a time-weighted method ultimately to calculate the popularity of task j.

$$Pop(j) = \alpha(t_s, t_c) * Pop_c(j) + \beta(t_s, t_c) * Pop_e(j) \quad (2)$$

Both $\alpha(t_s, t_c)$ and $\beta(t_s, t_c)$ are two time weighting factors, calculated by the task's release time t_c and the current time t_s. For $\alpha(t_s, t_c)$, it should be increased with $t_c - t_s$ since the reliability of current popularity is growing higher as released time goes by. Meanwhile, the reliability of estimated popularity decreases. So $\alpha(t_s, t_c)$ and $\beta(t_s, t_c)$ are defined as:

$$\alpha(t_s, t_c) = ln(t_c - t_s + 1), \quad \beta(t_s, t_c) = \frac{1}{ln(t_c - t_s + 1)} \quad (3)$$

B. Developer Modeling

We build the model of developers from three aspects.

1) Developer Profiling: When registering on different software crowdsourcing platforms, a new developer is required to fill in different individual information that can be converted into a unified vector to denote his profile model. The individual information generally contains three kinds of data type: numeric, string, and enumerations. Numeric data can be added directly to the profile vector, as well as enumeration after relatively simple discrete processing. For a string, it is generally accepted by converting it to a BOW vector so that the string can be appended to the profile vector.

2) Calculating Developer Activity Percentile: The developer activity percentile is calculated by his/her online behaviors, just according to the definition 1.

3) Refining Developer Topic: The relationship between developer and task is denoted by triples, representing whether a developer has bid, won, or delivered a task. In previous work [10] [11], only the bidding relation triples are used. In this paper, we use all relations to build developer topic model to keep more valuable information:

$$vu = \lambda_B \sum_{t \in B_u} v_t + \lambda_W \sum_{t \in W_u} v_t + \lambda_D \sum_{t \in D_u} v_t \tag{4}$$

where λ_B, λ_W and λ_D represent weights of bidding, winning and delivering, while B_u, W_u and D_u means sets of the task that developer u has bid, won and delivered. Delivering of the task reflects one developer's ability with the maximum influence, followed by winning and bidding, and thus we have $\lambda_B < \lambda_W < \lambda_D$. The more the tasks are delivered, won and bid, the stronger developer's ability is.

V. Cold-Start Developer Recommendation

In this section, we introduce a semi-supervised topic sampling algorithm to recommend tasks to cold-start developers on the software crowdsourcing platform. The algorithm consists of three steps: warm developer clustering, cold developer classification and recommendation by topic sampling.

A. Warm Developer Clustering

In the crowdsourcing platform, it is time-consuming to predefine the classes for all developers, which requires massive expert knowledge. In this paper, we apply clustering methods to obtain the classes automatically. Our algorithm only clusters warm developers because they have enough information left in the platform. K-means and DBScan are the effective clustering algorithms. We will compare the performance of these two algorithms in the experiments. The better one will be chosen.

After clustering by developer profiles, the cluster topic for each cluster is calculated by the formula-(5), where d is the number of developers in the cluster and $u_{i.topic}$ denotes the developer i's topic in the cluster. Thus it can be seen the cluster topic is the topic vector center of instances in one cluster.

$$Topic\,(C) = \frac{\sum_{i=1}^{d} u_{i.topic}}{d} \tag{5}$$

B. Cold-Start Developer Classification

For cold-start developers, they only have profile vector, and can be classified to warm developer clusters. To find a proper cluster for a cold-start developer, we first define cluster center for each cluster. Supposing there are d points in the cluster C, the cluster center is defined by formula-(6). Then for a cold-start developer, the distances between his profile vector and all cluster centers are calculated. The nearest cluster is the target cluster for the developer.

$$Center\,(C) = \frac{\sum_{i=1}^{d} u_{i.profile}}{d} \tag{6}$$

C. Recommendation by Topic Sampling

Accordingly, we design a recommendation algorithm based on topic sampling to make a recommendation for cold-start developers who are classified in a proper cluster.

1) Multi-objects Optimization Problem Formulation: From the profile aspect, a cold-start developer u has high similarity to developers in the same cluster. We consider that topic which the developer interests also has high similarity with those developers. However, the cold-start developer does not have the topic vector, so we use the corresponding cluster topic to represent the interested topic of the cold-start developer, denoted by $Topic_u\,(C)$. Based on $Topic_u\,(C)$, our algorithm samples l tasks which have high similarity to $Topic_u\,(C)$ and then recommends these tasks to the cold-start developer.

However, sometimes the assumption that the cluster topic can represent the cold-start developer's topic is not exactly precise since we only use developer profile to cluster and classify cold-start developer. To solve this issue, during the sampling process, we also take the task popularity into consideration.

Therefore, the cold-start developer recommendation is designed as a multi-objects optimization problem (MOP) that considers both task topic and popularity, formulated as

$$max. \frac{\sum_{i=1}^{l} Pop\,(t_i)}{l} \quad s.t. \frac{\sum_{i=1}^{l} t_{i.topic}}{l} - Topic\,(C) \leqslant \varepsilon \tag{7}$$

The constraint, $\frac{\sum_{i=1}^{l} t_{i.topic}}{l} - Topic\,(C) \leqslant \varepsilon$, tries to make the topic center of selected tasks close to the cluster topic $Topic_u\,(C)$. The optimization function $\frac{\sum_{i=1}^{l} Pop(t_i)}{l}$ ensures the overall popularity of selected tasks as high as possible. In our approach, task popularity considers both current popularity and popularity-to-be, so the recommendation algorithm can also recommend the latest tasks to developers, outperforming other popularity-based recommendation algorithms [1] [2].

2) Recommendation Algorithm Design: As an MOP, our recommendation problem is suitable to adopt genetic algorithm (GA) to search for optimal goals. GA is randomized search and optimization techniques based on the concepts of natural activity percentile of genes, individual selection, and the evolutionary process [20]. Applying the GA to the recommendation algorithm, the individual is the recommended task list, the gene is the task in the list and the population is the list of recommended lists. The optimization function, formula-(7), dictates that the next generation population will knock out

the list with a lower popularity and retain higher. The details of our recommendation algorithm are described as follows.

Algorithm: Topic Sampling Based Recommendation Algorithm

Input: all available tasks T
population size s
recommended tasks size l
cluster topic vector $Topic(C)$
selection criteria ε
iterative times t
mute rate α

Output: Rec_{list} [0]

1: Rec_{list} = **new** array(s)
2: **for** i **from** 1 **to** s:
3: Rec_{list}.append($new_instance$(Topic(C), ε, T, l)
4: Sort(Rec_{list})
5: **for** i **from** 0 **to** t:
6: **for** j **from** $\frac{s}{2}$ **to** s:
7: **if** $get_operation(\alpha)$=="mute" :
8: Rec_{list}[j] = mute(Rec_{list}[j], Topic(C), ε
9: **else**:
10: idx = $random_{int}$(0, $\frac{s}{2}$)
11: Rec_{list}[j], Rec_{list}[j+1]=cross(Rec_{list}[j], idx, Topic(C), ε)
12: j++
13: Sort(Rec_{list})

The first three lines of the algorithm initialize the initial population of random processing. Each individual is composed of two parts, gene and health degree, where the gene is the task list, the health degree is calculated using the formula (7). Then the algorithm sorts the health degree of the individual in descending order. The individuals with small health degrees in the current population are subject to mutation and cross-swapping operations (Line 6). Then the $get_{operation}$ function decides whether to perform mutation operations (Line 8) or cross-swaps (Lines 10-12) on the current individual referring to the mutation rate (Lines 7-9). Additionally, the objects of the cross-swapping are randomly selected from the high-quality individuals retained in the previous population (line 10). At the end, the most healthy individuals of the final population are returned.

In order to prevent it taking too much time in the mutate and cross functions, we can set a maximum number of mutations and the number of crossovers. When the number of mutations is greater than any of them, a new individual should be regenerated and returned.

VI. Evaluations

In order to evaluate the proposed cold-start recommendation approach, we have carried on several comparative experiments to answer four research questions.

RQ1: During task popularity estimation, which regression algorithm performs best?

RQ2: For the warm developer classification, which performs better, k-means or DBScan?

RQ3: During the task modeling, which popularity metric will more benefit the final recommendation result, Pop_c or Pop?

RQ4: Compared with other state-of-the-art recommendation approaches, how is the performance of our approach?

A. Experimental Dataset

We crawled data between Dec. 2010 to Apr. 2016 from Zhubajie, Oschina, and Witkey, which are all the largest software crowdsourcing platforms in China. There are several task categories in these platforms, and only software development tasks are considered in the experiments. In total, there are 15,375 developers and 31,255 software development tasks, as shown in Table I. Obviously, the data of Witkey is less than the other two platforms, and the relationship is relatively sparse.

TABLE I: Dataset from Three Software Crowdsourcing Platforms in China

Platfrom	Tasks	Developers	Relationship
Zhubajie	6000	10597	48769
Witkey	2800	5231	15498
Oschina	6575	15427	77925

B. Regression Experiment and Results

To answer the RQ1, we conducted the experiment to estimate task popularity by four widely used regression methods, including linear regression, classification and regression tree (CART), Gaussian mixture and artificial neural network. And three well-known metrics are used to evaluate predictive accuracy: mean absolute error (MAE), root mean square error (RMSE) and R-square.

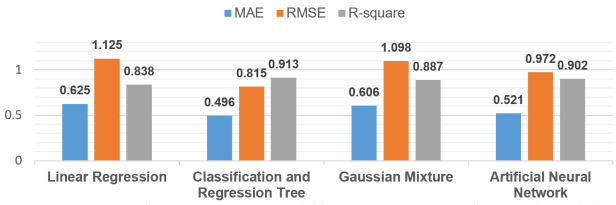


Fig. 2: Comparison among Different Regression Algorithms.

Fig. 2 shows the results, where CART achieves the best performance with the lowest MAE, RMSE and the highest R-square. Thus in our approach, we apply the CART to estimate task popularity.

C. Clustering Experiment and Results

To answer RQ2, we conducted a cold-start developer recommendation experiment using the k-means algorithm ($k = 15$ in the experiments) and DBScan algorithm for warm developer clustering in our approach. We use precision and recall as the evaluation metrics.

We experimentally set λ_B, λ_W and λ_D to 0.3, 0.8 and 1.0 in formula (4), respectively. Considering the requirements from two real software crowdsourcing platforms, we set recommended list size to 15, i.e., $l = 15$. The other settings in our algorithm are: (1) the popularity size s is 100; (2) the iterative times t is 100; (3) the mute rate α is 0.3.

TABLE II: Comparison between Two Clustering Algorithms

Platform	K-means		DBScan	
	Precision	Recall	Precision	Recall
Zhubajie	**0.099**	**0.557**	0.087	0.524
Witkey	0.069	0.531	**0.072**	**0.559**
Oschina	**0.077**	**0.481**	0.037	0.361

TABLE III: Cold-Start Developer Recommending Results

Platform	Ours with Pop_c		Ours with Pop		NRec with Pop_c		NRec with Pop		MRec with Pop_c		MRec with Pop	
	Precision	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
Zhubajie	0.073	0.531	**0.099**	**0.557**	0.052	0.317	0.068	0.433	0.059	0.346	0.079	0.474
Witkey	0.059	0.498	**0.069**	**0.531**	0.044	0.286	0.047	0.398	0.049	0.313	0.054	0.442
Oschina	0.066	0.449	**0.077**	**0.481**	0.035	0.243	0.046	0.354	0.040	0.269	0.057	0.396

The recommendation results using two clustering algorithms are listed in Table II, which show that k-means wins on stability. In the dataset of Oschina, DBScan results in the worst recommendation recall and precision, because DBScan clusters all warm developers as one group.

D. Recommendation Experiment and Results

To answer the RQ3 and RQ4, we select two popular popularity-based recommendation approaches in [10] and [11], might called NRec and MRec, to perform the task recommendation for the cold start developers, and compare them with our approach. Each approach will adopt the two popularity metrics respectively, i.e. Pop_c and Pop. We also use precision and recall to evaluate the recommendation results.

The experimental results are shown in Table III. It can be observed that our approach outperforms MRec and NRec either with Pop_c or Pop, and improves 75% of precision and recall on average.

Even though the recommendation results of NRec and MRec with Pop are not the best one, they performs better than with Pop_c (i.e. the current popularity). Also MRec with Pop_c and NRec with Pop_c cannot recommend the latest requested task. So it is obvious that Pop is more beneficial to the final recommendation result than Pop_c.

VII. Conclusion and Future Work

In this paper, we propose a novel approach to task recommendation for cold-start developers in crowdsourcing platforms, using a topic-sampling based genetic algorithm. It builds the model of cold-start developers by leveraging the warm developers' preference and tasks' topics. Experimental results show that our approach can minimize the impact of the developer cold-start problem effectively.

For the future work, we plan to explore the task cold-start problem in depth, i.e., recommend developers to cold-start tasks, which is equally important. We will also research on the platform cold-start problem by transfer learning technologies, to build an effective recommender system for a new software crowdsourcing platform.

Acknowledgement

This research is supported by 973 Program in China (Grant No. 2015CB352203) and National Natural Science Foundation of China (Grant No. 61472242 and 61572312).

References

[1] K. Christakopoulou, F. Radlinski, and K. Hofmann, "Towards conversational recommender systems," in *The ACM SIGKDD International Conference*, pp. 815–824, 2016.

[2] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 253–260, 2002.

[3] S. Li, A. Karatzoglou, and C. Gentile, "Collaborative filtering bandits," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 539–548, 2016.

[4] O. N. Osmanli and I. H. Toroslu, "Using tag similarity in svd-based recommendation systems," in *International Conference on Application of Information and Communication Technologies*, pp. 1–4, 2011.

[5] Tak, G. Cs, I. Szy, B. Meth, and D. Tikk, "Matrix factorization and neighbor based algorithms for the netflix prize problem," in *ACM Conference on Recommender Systems (Recsys), Lausanne, Switzerland*, pp. 267–274, 2008.

[6] N. Golbandi, Y. Koren, and R. Lempel, "On bootstrapping recommender systems," in *ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October*, pp. 1805–1808, 2010.

[7] A. M. Rashid, G. Karypis, and J. Riedl, "Learning preferences of new users in recommender systems: an information theoretic approach," *Acm Sigkdd Explorations Newsletter*, vol. 10, no. 2, pp. 90–100, 2008.

[8] F. Hu and Y. Yu, "Interview process learning for top-n recommendation," in *ACM Conference on Recommender Systems*, pp. 331–334, 2013.

[9] R. Karimi, A. Nanopoulos, and L. Schmidt-Thieme, "Improved questionnaire trees for active learning in recommender systems," *Proceedings of the LWA 2014 Workshops*, pp. 6–11, 2014.

[10] Z. Miao, J. Yan, K. Chen, X. Yang, H. Zha, and W. Zhang, "Joint prediction of rating and popularity for cold-start item by sentinel user selection," *IEEE Access*, vol. 4, pp. 8500–8513, 2016.

[11] I. Arapakis, B. B. Cambazoglu, and M. Lalmas, "On the feasibility of predicting news popularity at cold start," in *6th International Conference on Social Informatics, Barcelona, Spain*, pp. 290–299, November 11-13, 2014.

[12] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *International Conference on World Wide Web*, pp. 661–670, 2010.

[13] J. Zhu, B. Shen, and F. Hu, "A learning to rank framework for developer recommendation in software crowdsourcing," in *Asia-Pacific Software Engineering Conference*, pp. 285–292, 2015.

[14] S. Lee, S. Park, and S. Park, "A quality enhancement of crowdsourcing based on quality evaluation and user-level task assignment framework," in *International Conference on Big Data and Smart Computing*, pp. 60–65, 2014.

[15] M. C. Yuen, I. King, and K. S. Leung, "Task recommendation in crowdsourcing systems," in *International Workshop on Crowdsourcing and Data Mining*, pp. 22–26, 2012.

[16] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[17] M. Khandelwal, D. J. Armaghani, R. S. Faradonbeh, M. Yellishetty, M. Z. A. Majid, and M. Monjezi, "Classification and regression tree technique in estimating peak particle velocity caused by blasting," *Engineering with Computers*, vol. 32, no. 120, pp. 1–9, 2016.

[18] B. Pavlyshenko, "Machine learning, linear and bayesian models for logistic regression in failure detection problems," in *IEEE International Conference on Big Data (BigData), Washington DC, USA*, pp. 2046–2050, December 5-8, 2016.

[19] J. Ala-Luhtala and R. Piché, "Gaussian scale mixture models for robust linear multivariate regression with missing data," *Communications in Statistics - Simulation and Computation*, vol. 45, no. 3, pp. 791–813, 2016.

[20] A. H. Beg and M. Z. Islam, "A novel genetic algorithm-based clustering technique and its suitability for knowledge discovery from a brain data set," in *IEEE Congress on Evolutionary Computation*, pp. 948–956, 2016.

Automated Software Security Requirements Recommendation Based on FT-SR Model

Jiangjuan Wang[1,3], Xiaohong Li[1,3], Zhiyong Feng[2,3], Jianye Hao[2,3], Guangquan Xu[1,3], Zhuobing Han[1,3]
[1]School of Computer Science and Technology, Tianjin University,
[2]School of Computer Software, Tianjin University,
[3]Tianjin Key Laboratory of Advanced Networking, Tianjin, China
Email:{jiangjuanwang, xiaohongli, zyfeng, jianye.hao, losin, zhuobinghan}@tju.edu.cn

***Abstract*—Since security is recommended to be evaluated at the beginning of the software development process, specifying software security requirements is inevitable in developing Critical Information Security Systems. However, according to the ISO/IEC 15408 (known as Common Criteria), determining detailed software security requirements (SRs) is quite challenging, complex which needs lots of expert knowledge of security. In this paper, a data-driven Functionality Topic-Security Requirement (FT-SR) model is proposed to recommend software SRs based on the relationship between software functionality specification and SRs from software Security Target (ST) which have been written in accordance with ISO/IEC 15408. First, we extract descriptions of functionality and tag SRs all from software STs. Second, Latent Dirichlet Allocation (LDA) is adopted to build functionality topics for product functionality description. Third, a FT-SR model is developed based on the mapping between product functionality topics and SRs which have been tagged in ST documents. Finally, a recommendation strategy is proposed to recommend SRs based on the FT-SR model for software products. Our experiments are performed on ST documents of over 600 software products provided by Common Criteria. Experimental results show that the proposed approach can generate a set of recommended SRs reducing the difficulty of SRs recommending even for people lack knowledge of security.**

Keywords-Security Requirements; the Common Criteria; Software Functional Description; FT-SR Model

I. Introduction

Many security issues occur in software as a result of errors and misspecifications in analysis, design and implementation [1]. Nowadays developers are more concerned about software security in the software development process. In order to develop security software products, specifying SRs is vitally important [2]. Security requirements of software systems can be challenging to identify during the requirements engineering process [3]. So far many methods have been proposed for SRs elicitation reference. Early work suggests that use case has been widely used during requirements for eliciting security threats and requirements [4]. As a result, use cases [4], misuse cases [5] and abuse cases [6] have all been proposed for specifying SRs, respectively. Although these methods are useful, they are inefficient and time-consuming because of without the guidance. Maria Riaz, et al. [7] developed a tool-assisted process presenting a list of applicable SRs templates for identifying SRs. And, they prove that the automatically-suggested templates usage increases the efficiency of identifying security requirements by comparing to a manual approach without the guidance of templates [3]. These researches all need using standards such as Common Criteria (referred to as "CC" in this paper), known as ISO/IEC 15408 [8]–[10]. The standard allows to select suitable SRs for a specific product or a group of products. However, since CC standards are too confusing to understand, these researches about determining softwares SRs processes are difficult and complex.

In this paper, we propose a new research direction of identifying and ensuring SRs for software products which only have functional descriptions without SRs. We firstly extract text description related to software functionality and tag related SRs all from software ST documents. ST documents are written in accordance with CC and certified by Competent and Independent Licensed Laboratories. Second, Latent Dirichlet Allocation (LDA) is used to build functionality topic feature for functional description of the evaluated software products. Meanwhile, according to the relationship between software functionality and SRs showing in STs, a FT-SR model is generated for software products. Finally, for given software products which only have functional description, SRs are recommended on the basis of the FT-SR model. A recommended strategy combining with a filtering framework [11] is developed to recommend SRs.

Therefore, the main contribution of this paper is as follows:

- A novel method is proposed to obtain softwares SRs even if person has little knowledge of security.
- The Functionality Topic-Security Requirement (FT-SR) model is built based on the mapping between functional description of software products and software products security requirements.
- A data-driven measure based on text description is considered to solve software security problems.

The remainder of the paper is organized as follows. Section 2 describes the methodology of overview. Section 3 analyzes the proposed method conducted in detail. Section 4 presents the experimental results and discussion. Section 5 concludes the paper and points out future work.

II. Method Overview

A. Problem Definition

As paying attention to software SRs and the relationship between functional description and SRs, we can formally

DOI reference number: 10.18293/SEKE2017-040

define SRs and the relationship respectively in order like:

$$SR_i = \{SFR_i, SAR_i\} = \{< sfc_{i,1}, \ldots, sfc_{i,K} >, < sac_{i,1}, \ldots, sac_{i,P} >\} \quad (1)$$

Note that, CC defines SRs as two aspects: security functional requirements (SFR) [9] and security assurance requirements (SAR) [10]. SFR_i and SAR_i respectively refer to security functional requirements and security assurance requirements which are SRs SR_i of software product S_i needed. $sfc_{i,j}$ represents security functional component. $sac_{i,j}$ represents security assurance component. K is the number of security functional components and P is the number of security assurance components in Common Criteria.

$$s_i = \{F_i, SR_i\} = \{< tp_{i,1}, \ldots, tp_{i,N} >, < sfc_{i,1}, \ldots, sfc_{i,K} >, < sac_{i,1}, \ldots, sac_{i,P} >\} \quad (2)$$

Note that: $F_i =< tp_{i,1}, \ldots, tp_{i,N} >$ refers to functionality feature of software product s_i , N is the total number of functionality topic for all software products, $0 \leq tp_{i,j} \leq 1$ represents the probability that software functionality is relevant to a certain functionality topic.

Accordingly, the problem of security requirements based on software functionality topic are defined as:

Q1: given the software product s_i and functional description from STs, how to generate the functionality topic feature and recommend its security requirements?

Q2: given the software product s_i, functional description and the recommendation security requirements, how to evaluate the recommendation results in our experiment?

B. Framework Overview

In order to solve those questions mentioned above, we introduce the architecture of our method which combines the FT-SR model and a recommendation strategy to recommend SRs for software products only having functionality requirements without SRs. It is shown in Fig.1.

Software Functional Description Gathering and Security Requirements Tagging: The goal of this process is to gather functional description and tag SRs from ST documents. Detail will be shown in Section 3.1.

Software Functionality Topic Clustering: The goal for this process is to obtain software functionality topic feature for all software products based on their functional descriptions. Here, we use LDA to cluster functional description being gathered and generate functionality topic features. Detail will be shown in Section 3.2.

Software Functionality Topics and Security Requirements (FT-SR) Model Building: For each software product, we have tagged its security requirements in accordance with its ST. We build functional topic-security requirements (FT-SR) model based on the mapping between its functional topics and its SRs. Detail will be shown in Section 3.3.

Security Requirements Recommending: For software product s_i which only have functionality requirements, we can get its SRs based on the functionality topics feature and FT-SR model combining a recommendation strategy. Detail will be shown in Section 3.4.

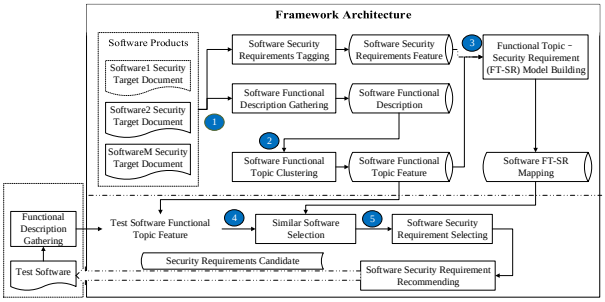


Fig. 1: Architecture of FTM-based Security Requirements Recommendation

III. The Proposed Method

This section discusses the methodology that has been used in the work.

A. Software Functional Description Gathering and Security Requirements Tagging

In this section, the descriptions of software functionality are extracted and security requirements are tagged all from Security Target for software products. Those software products are publicly available on the Common Criteria (http://www.commoncriteriaportal.org/products/). The ST documents are involved in more than 1000 products which have been certified by Competent and Independent Licensed Laboratories. ST documents have been downloaded and collected in this paper. Our task is to carefully read software product's ST documents to extract the description of functionality and tag SRs. For each software product, we extract its functional descriptions from the Chapter 1 of ST documents, meanwhile, we tag its security requirements in accordance with the Chapter 6 of ST.

B. Software Functional Topic Clustering

The goal for this process is to generate functionality topics for all the software products based on their functional descriptions. We follow the methodology using the Latent Dirichlet Allocation (LDA) to clustering all software's functional descriptions into different functionality topics, so that we can get the probability that each software product is relevant to certain functionality topics. Note that the summation of all the elements for each topic feature vector equals to 1. The results of LDA are N functional topics and the probability that each software product is relevant to the N functionality topics. Thus, functionality of all software products is represented as a topic feature vector, as follows:

$$F = \begin{bmatrix} tp_{1,1} & tp_{1,2} & \ldots & tp_{1,N} \\ tp_{2,1} & tp_{2,1} & \ldots & tp_{2,N} \\ \ldots & \ldots & \ldots & \ldots \\ tp_{M,1} & tp_{M,1} & \ldots & tp_{M,N} \end{bmatrix} \quad (3)$$

Where: $tp_{i,j}$ refers to the probability that one software product is relevant to one functional topic, $0 \leq tp_{i,j} \leq 1$. $F_i =< tp_{i,1}, \ldots, tp_{i,N} >$ refers to the set of probability that software product s_i is relevant to each functional topics. N is the number of all functionality topics that all software products were clustered into different functionality topics; M is the number of all software products.

In this section, the correlation which also is functionality topic feature between functional description and each functionality topic can be obtained for each software product.

C. Functional Topic-Security Requirements (FT-SR) Model

For each software product, SRs are tagged in section 3.1. And, the FT-SR model is built between the relationship between functionality topics feature, software product and SRs for software products. First, the tagged SRs feature is presented a feature vector, as follow:

$$SR = \begin{bmatrix} sfc_{1,1} & \ldots & sfc_{1,K}, & sac_{1,1} & \ldots & sac_{1,P} \\ sfc_{2,1} & \ldots & sfc_{2,K}, & sac_{2,1} & \ldots & sac_{2,P} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ sfc_{M,1} & \ldots & sfc_{M,K}, & sac_{M,1} & \ldots & sac_{M,P} \end{bmatrix} \quad (4)$$

Where $sfc_{i,j}$ represents specific security functional requirements sfc_j what software product s_i needed. K is the total security functional components what CC have defined. Note that, if $sfc_{i,j} = 0$, this means the software product s_i doesn't use security functional component sfc_j.

Where $sac_{i,j}$ represents the specific security assurance requirements sac_j what software product s_i needed. P is the total security assurance components what CC have defined. Note that, if $sac_{i,j} = 0$, this means the software product s_i doesn't use security assurance component sac_j.

According to above steps, we build the FT-SR model which is the relationship between functionality topic and SRs for each software product. The relationship is expressed as same as (2).

D. Security Requirements Recommending

Software Products of Similar Functionalities Selecting.

For a given software product s_i which have only functional descriptions, as LDA is used to mapping its functional description into generate probability of each functionality topics, software product s_i is represented as a functionality topic feature vector. Therefore, this paper calculate the similarity between two software products s_i, s_j as follow:

$$Sim(s_i, s_j) = \frac{\sum_{n=1}^{N} tp_{i,n} \times tp_{j,n}}{\sqrt{(\sum_{n=1}^{N} tp_{i,n}^2)(\sum_{n=1}^{N} tp_{j,n}^2)}} \quad (5)$$

Thus, we can get a list which related software products with similarity functionality to the given software product s_i .Using a subset of software products for recommendation can gain a better performance because of the noise in the dataset. Therefore, we use the threshold method to obtain a subset of appropriate software products with similarity larger than the given threshold θ_{sim}. Given the selected software products $P^*(s_i)$ with similarity larger than the given threshold θ_{sim}. The threshold is changing and the better threshold has been found by comparing the assurance of recommendation result in our experiment.

Security Requirements Selecting and Recommending.

According to the selected software products $P^*(s_i)$, we can calculate the likelihood that software product s_i need security requirements SR_j is calculated as follow:

$$l(s_i, SR_{i,j}) = \frac{\sum_{s_k \in P^*(s_i)} Sim(s_i, s_k) l(s_k, SR_{k,j})}{\sum_{s_k \in P^*(s_i)} Sim(s_i, s_k)} \quad (6)$$

Note that, if $l(s_k, SR_{k,j})$=1 , software product s_k needs the security requirement SR_j, otherwise $l(s_k, SR_{k,j})$=0. Thus, we select the likelihood vector for the given software products $L(s_i) =< l(s_i, sfc_{i,1}), \ldots, l(s_i, sfc_{i,Q}), l(s_i, sac_{i,1}), \ldots, l(s_i, sac_{i,R}) >$, where $l(s_i, sfc_{i,j}) \geq l(s_i, sfc_{i,k}) \geq \theta_{NP}$,$1 \leq j \leq k \leq Q$, $l(s_i, sac_{i,j}) \geq l(s_i, sac_{i,k}) \geq \theta_{NP}$, $1 \leq j \leq k \leq R$, θ_{NP} is the likelihood threshold.

Therefore, the set of $SR_i =< sfc_{i,1}, \ldots, sfc_{i,Q}, sac_{i,1}, \ldots, sac_{i,R} >$ is the recommendation of security requirements candidate for software product s_i.

IV. Experimental Results and Evaluation

A. Data Set

Since Common Criteria is well-known for Information Technology Security Evaluation and most software products are evaluated by Common Criteria, we use the "Certified Products" as dataset from http://www.commoncriteriaportal.org/products/. The software products have been classified into 14 categories by Competent and Independent Licensed Laboratories. Our method applies to the 14 categories of software products which behave in the exact similar way. Considering the number of software products each category, we have mainly gathered three categories of products which are ICs, Smart Cards and Smart Card-Related Devices and Systems, Multi-Function Devices and Network and Network-Related Devices and Systems because of their large quantity. Then as software products have grown into different versions, we choose the latest version of every product and download its Security Target separately in accordance with categories. Overview of datasets is shown in Table 1.

TABLE I: Overview of Datasets

Dataset	Number
ICs, Smart Cards and Smart Card-Related Devices and Systems	403
Multi-Function Devices	115
Network and Network-Related Devices and Systems	138

B. Result and Evaluation

In our method, we use the similarity threshold θ_{sim} to select the software products which have similar functionalities. Therefore, in order to evaluate its assurance, we randomly select 25% software products from each category products as the testing dataset respectively and the three experiments

reported in Figure 2. We vary threshold θ_{sim} from 0 to 1 to generate different recommendations. Mean Average Precision (MAP) are used to evaluate the performance of accuracy for results of recommendation and answer the question 2 mentioned above. It can be formally defined as follow:

$$MAP = \frac{1}{R}\sum_{r=1}^{R}\frac{I_r}{N_r} \tag{7}$$

Where R refers to the number of software products in testing dataset, N_r refers to the number actually used security requirements for rth product s_r. I_r refers to the number of the recommended security requirements which belong to the actually used security requirements N_r. And, the MAP of the recommendation is reported in Fig 2. As shown in Fig 2, it can be seen that for the dataset, if $\theta_{sim} > 0.5$, the MAP is decreasing; This is because that with a too larger similarity threshold, most of software products can not have enough similar products to generate a valid recommendation. Therefore, in the rest of this paper, we set $\theta_{sim} = 0.5$ for the selection. For the dataset, those software products are classified into 14 categories. Thus, for the given software products, its type should also belong to the 14 categories for recommendations.

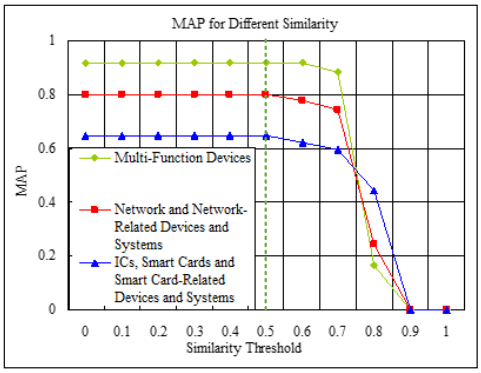


Fig. 2: Overview of Datasets

C. Comparison

Now, Common Criteria is used to analyze security requirements internationally. This process specifies [12], among other things, (a) which are the assets to be protected (b) which are the threats to the declared assets and (d) which are the countermeasures. The CC then describes the countermeasures and demonstrates that they are sufficient to counter the declared threats. Therefore, if developers want to get SRs for software products, they have to acquire the knowledge of CC and find out what kind of information they needed in the procedures. So according to Common Criteria security standard, obtaining softwares SRs process is an extremely difficult and complex task. In addition, its results are largely influenced by subjective knowledge of analyst, so accuracy of security requirements can't be guaranteed.

However, our method solves those disadvantages showing above. For given software products which only have functional requirements without SRs, we could recommend its SRs applying on their functional descriptions even for people with little knowledge of security engineering and Common Criteria.

V. Conclusion

Identifying SRs is an important part during the software development. Most of existing work is related to Common Criteria. Since Common Criteria standards are often too confusing to understand, these researches are difficult and complex. In this paper, functionality of software products are considered to recommend SRs. Functional descriptions are extracted manually and clustered into functionality topic. Basing on the mapping between SRs and functionality topics, the FT-SR model is built. Finally, for given software products which only have functional requirements without SRs, their SRs are recommended based on FT-SR model. The advantage of our method is acquiring SRs even for people lack knowledge about security.

In the future, we will further extend our method to more precisely recommend SRs of software products to help SRs identification. We can consider more software information to recommend more appropriate SRs.

VI. Acknowledgement

This work has partially been sponsored by the National Science Foundation of China (No. 61572349, 61272106).

References

[1] U. Erlingsson, "Data-driven software security: Models and methods," in *Computer Security Foundations Symposium (CSF), 2016 IEEE 29th*. IEEE, 2016, pp. 9–15.

[2] D. Mellado, E. Fernández-Medina, and M. Piattini, "A common criteria based security requirements engineering process for the development of secure information systems," *Computer standards & interfaces*, vol. 29, no. 2, pp. 244–253, 2007.

[3] M. Riaz, J. Slankas, J. King, and L. Williams, "Using templates to elicit implied security requirements from functional requirements-a controlled experiment," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 2014, p. 22.

[4] M. S. Ware, J. B. Bowles, and C. M. Eastman, "Using the common criteria to elicit security requirements with use cases," in *SoutheastCon, 2006. Proceedings of the IEEE*. IEEE, 2005, pp. 273–278.

[5] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements engineering*, vol. 10, no. 1, pp. 34–44, 2005.

[6] J. Mcdermott, "Abuse-case-based assurance arguments," in *Computer Security Applications Conference, 2001. ACSAC 2001. Proceedings*, 2001, pp. 366–374.

[7] M. Riaz, J. King, J. Slankas, and L. Williams, "Hidden in plain sight: Automatically identifying security requirements from natural language artifacts," in *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*. IEEE, 2014, pp. 183–192.

[8] "Common criteria for information technology security evaluation (version 3.1, revision 4) part 1: Introduction and general model," *ISO/IEC 15408*, 2012.

[9] "Common criteria for information technology security evaluation (version 3.1, revision 4) part 2: Security functional components," *ISO/IEC 15408*, 2012.

[10] "Common criteria for information technology security evaluation (version 3.1, revision 4) part 3: Security assurance components," *ISO/IEC 15408*, 2012.

[11] K. Huang, J. Han, S. Chen, and Z. Feng, *A Skewness-Based Framework for Mobile App Permission Recommendation and Risk Evaluation*. Springer International Publishing, 2016.

[12] M. Saeki and H. Kaiya, *Security Requirements Elicitation Using Method Weaving and Common Criteria*. Springer Berlin Heidelberg, 2009.

A Reinforced Hungarian Algorithm for Task Allocation in Global Software Development

Xiao Yu
State Key Lab. of Software Engineering,
Computer School, Wuhan University,
Wuhan, China
xiaoyu_whu@yahoo.com

Man Wu
School of Computer Science and Information Engineering,
HuBei University,
Wuhan, China
wuman_1@qq.com

Xiangyang Jia*
State Key Lab. of Software Engineering,
Computer School, Wuhan University,
Wuhan, China
Corresponding author email: jxy@whu.edu.cn

Ye Liu
School of Computer Science and Information Engineering,
HuBei University,
Wuhan, China
liuye061@qq.com

***Abstract*—The allocation of software development tasks is a critical management activity in distributed development projects. One of the most important problem is to find the lowest-cost way to assign tasks in global software development, which can be solved by Hungarian algorithm. However, the original Hungarian algorithm only assume that a task can only be solved by one development site. The assumption is not agreed with the actual case where a software development task is usually be solved through a collaboration among several sites. To address such an issue, this paper proposes a reinforced Hungarian algorithm (RHA) for task assignment in global software development. RHA consists of three major stages. First, RHA transforms a *n*×*m* cost matrix into two *n*×*n* cost matrix by adding (2*n*-*m*) virtual development sites. Second, RHA performs the original Hungarian algorithm on the two *n*×*n* cost matrix to get the optimal assignment results. Finally, RHA removes the (2*n*-*m*) virtual development sites and gets the final optimal assignment result for *m* tasks. Simulation results indicate that RHA is a viable approach for the task assignment problem in global software development.**

Keywords—task allocation; global software development; Hungarian algorithm

I. INTRODUCTION

As global and distributed software development is becoming a norm in the software industry [1-2], a tight budget and a shortage of resources and time have motivated many software enterprises to start looking for outside partners. With the advent of big data era [3-4] and the development of network technology [5-6], known as global software development (GSD), the allocation of software development tasks over several sites, which may be spread across different countries, has become a common practice in industrial software engineering [7-8].

DOI reference number: 10.18293/SEKE2017-042

Software development task assignment is one of the core steps to reasonably allocate limited development resources and effectively exploit the capabilities of the development sites in global software development. However, task allocation is still one of the major challenges in global software development.

A large number of possible combinations for task assignment makes the process more complicated. For example, if a software development project consists of five different tasks which need to be assigned among four development sites, theoretically there are 4^5=1024 different combinations available for task assignment [9]. It is evident that evaluating all possibilities is impossible and it needs an approach to assign the task to the available development sites.

There are several possible strategies for allocating tasks that use different criteria for allocation.

Bass et.al [10] state that in practice, allocating task to low-cost countries has become a cost-saving strategy for many organizations. Therefore, one of the most important problem is to find the lowest-cost way to assign tasks in global software development, which usually be solved by Hungarian algorithm [11]. However, the original Hungarian algorithm solves the problem that the project manager allocates *n* tasks to *n* sites with the least total cost by performing all operations on a *n*×*n* cost matrix. It only assumes that the number of tasks is equal to the number of sites and a task can only be solved by one development site. The assumption is not agreed with the actual case where the number of tasks and the number of site are not equal in general, and a software development task is usually be solved through a collaboration among several sites.

To address such an issue, this paper proposes a reinforced Hungarian algorithm (RHA) for task assignment in global software development. The algorithm solves the problem that the project manager allocates *n* tasks to *m* sites at the least total cost by performing all operations on a *n*×*m* matrix. RHA

consists of three major stages. First, RHA transforms a $n \times m$ matrix into two $n \times n$ matrix by adding (m-n) virtual tasks. Second, RHA performs the original Hungarian algorithm on the two $n \times n$ cost matrix to get the optimal assignment for $2n$ tasks. Finally, RHA removes the (m-n) virtual tasks and gets the final optimal assignment for m tasks.

The effectiveness of RHA is demonstrated by comparing it with the expansion matrix method [12]. Simulation results indicate that RHA can get the optimal assignment with the lowest cost in global software development.

The remainder of this paper is organized as follows. Section II reviews the Hungarian algorithm. Section III describes the proposed reinforced Hungarian algorithm (RHA) for task assignment in global software development and Section IV shows the operational process of the algorithm by an example. Section V demonstrates the experimental results. Section VI presents the related work. Finally, Section VII addresses the conclusion and points out the future work.

II. HUNGARIAN ALGORITHM

In this section, we first give the problem definition. Then, we briefly review the Hungarian algorithm.

A. Problem definition

The usual assignment problem is defined as follows: assign n tasks to n development sites with the least total cost, if task i is assigned to site j with a non-negative integer cost c_{ij}. This problem is a special case of the linear programming problem, defined as follows:

$$\text{Min } S=\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \quad (1)$$

$$\text{Subject to } \sum_{i=1}^{n} x_{ij} = 1 \text{ (j=1, 2, ..., n)}, \quad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \text{ (i=1, 2, ..., n)}, \quad (3)$$

$$x_{ij}=1 \text{ or } 0. \quad (4)$$

where x_{ij}= 1 means task i is assigned to site j, otherwise x_{ij}=0, task i is not assigned to site j. The costs c_{ij} form a cost matrix, denoted C.

Theorem 1: If a number is added to or subtracted from all of the entries of any one row or column of a cost matrix C, then the optimal assignment for the resulting cost matrix C' is also an optimal assignment for the original cost matrix C.

Proof: Assume that u_i is subtracted from all of the entries in each row and v_j is subtracted from all of the entries in each column from the original cost matrix C. For the resulting cost matrix C', the problem is defined as follows:

$$\begin{aligned}\text{Min } S' &= \sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij}-u_i-v_j)\times x_{ij} \quad (5)\\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n} u_{ij}x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n} v_{ij}x_{ij}\end{aligned}$$

Because $\sum_{i=1}^{n} x_{ij} = 1$ (j=1,2,3…,n) and $\sum_{j=1}^{n} x_{ij} = 1$ (i=1,2,3…,n),

then

$$\begin{aligned}\text{Min } S' &= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\times x_{ij} - \sum_{i=1}^{n} u_i - \sum_{j=1}^{n} v_j\\ &= \text{Min } S - \sum_{i=1}^{n} u_i - \sum_{j=1}^{n} v_j\end{aligned}$$

Since $\sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j$ is a constant number, the optimal assignment for the resulting cost matrix C' is also an optimal assignment for the original cost matrix C

B. The Hungarian algorithm

The Hungarian algorithm applies the above mathematical model and theorem 1 to a given cost matrix to find an optimal assignment.

Step 1. Subtract the smallest entry in each row from all the entries of its row.

Step 2. Subtract the smallest entry in each column from all the entries of its column.

Step 3. Draw lines through appropriate rows and columns so that all the zero entries of the cost matrix are covered and the minimum number of such lines is used.

Step 4. Test for Optimality: (1) If the minimum number of covering lines is, an optimal assignment of zeros is possible and we are finished. (2)If the minimum number of covering lines is less than, an optimal assignment of zeros is not yet possible. In that case, proceed to Step 5.

Step 5. Determine the smallest entry not covered by any line. Subtract this entry from each uncovered row, and then add it to each covered column. Return to Step 3.

C. Example

A software development project consists of four different tasks: Task 1, Task 2, Task 3 and Task 4. There are four development sites: S1, S2, S3, and S4. They each demand different pay for various tasks. The problem is to find the lowest-cost way to assign the tasks. The 4×4 cost matrix of this problem is shown as follows.

$$\begin{bmatrix} 9.4 & 4.8 & 5 & 8.2 \\ 9.6 & 10 & 6.4 & 8.8 \\ 6.6 & 11 & 8 & 7.6 \\ 8.2 & 8.4 & 5 & 8.2 \end{bmatrix}$$

Step 1. Subtract 4.8 from Row 1, 6.4 from Row 2, 6.6 from Row 3, and 5 from Row 4. The resulting matrix is as follows.

$$\begin{bmatrix} 4.6 & 0 & 0.2 & 3.4 \\ 3.2 & 3.6 & 0 & 2.4 \\ 0 & 4.4 & 1.4 & 1 \\ 3.2 & 3.4 & 0 & 3.2 \end{bmatrix}$$

Step 2. Subtract 1 from Column 4. The resulting matrix is as follows.

$$\begin{bmatrix} 4.6 & 0 & 0.2 & 2.4 \\ 3.2 & 3.6 & 0 & 1.4 \\ 0 & 4.4 & 1.4 & 0 \\ 3.2 & 3.4 & 0 & 2.2 \end{bmatrix}$$

Step 3. Cover all the zeros of the matrix with the minimum number of horizontal or vertical lines.

$$\begin{bmatrix} \cancel{4.6} & \cancel{0} & \cancel{0.2} & \cancel{2.4} \\ 3.2 & 3.6 & 0 & 1.4 \\ \cancel{0} & \cancel{4.4} & \cancel{1.4} & \cancel{0} \\ 3.2 & 3.4 & 0 & 2.2 \end{bmatrix}$$

Step 4. Since the minimal number of lines is less than 4, we have to proceed to Step 5.

Step 5. Note that 1.4 is the smallest entry not covered by any line. Subtract 1.4 from each uncovered row. The resulting matrix is as follows.

$$\begin{bmatrix} 4.6 & 0 & 0.2 & 2.4 \\ 1.8 & 2.2 & -1.4 & 0 \\ 0 & 4.4 & 1.4 & 0 \\ 1.8 & 2 & -1.4 & 0.6 \end{bmatrix}$$

Now add 1.4 to each covered column. The resulting matrix is as follows.

$$\begin{bmatrix} 4.6 & 0 & 1.6 & 2.4 \\ 1.8 & 2.2 & 0 & 0 \\ 0 & 4.4 & 2.8 & 0 \\ 1.8 & 2 & 0 & 0.6 \end{bmatrix}$$

Now return to Step 3.

Step 3. Cover all the zeros of the matrix with the minimum number of horizontal or vertical lines.

$$\begin{bmatrix} \cancel{4.6} & \cancel{0} & \cancel{1.6} & \cancel{2.4} \\ \cancel{1.8} & \cancel{2.2} & \cancel{0} & \cancel{0} \\ \cancel{0} & \cancel{4.4} & \cancel{2.8} & \cancel{0} \\ \cancel{1.8} & \cancel{2} & \cancel{0} & \cancel{0.6} \end{bmatrix}$$

Step 4. Since the minimal number of lines is 4, an optimal assignment of zeros is possible and we are finished.

$$\begin{bmatrix} 4.6 & \boxed{0} & 1.6 & 2.4 \\ 1.8 & 2.2 & 0 & \boxed{0} \\ \boxed{0} & 4.4 & 2.8 & 0 \\ 1.8 & 2 & \boxed{0} & 0.6 \end{bmatrix}$$

Since the total cost for this assignment is 0, it must be an optimal assignment. Here is the same assignment made to the original cost matrix

$$\begin{bmatrix} 9.4 & \boxed{4.8} & 5 & 8.2 \\ 9.6 & 10 & 6.4 & \boxed{8.8} \\ \boxed{6.6} & 11 & 8 & 7.6 \\ 8.2 & 8.4 & \boxed{5} & 8.2 \end{bmatrix}$$

So we should assign Task 1 to Site 2, Task 2 to Site 4, Task 3 to Site 1, and Task 4 to Site 3.

III. RHA

In this section, we first give the problem definition. Then, we present our RHA method for task allocation in global software development.

A. Problem definition

The original Hungarian algorithm only assume that the number of tasks is equal to the number of sites and a task can only be solved by one development site and a task can only be solved by one development site. The assumption is not agreed with the actual case in global software development where the number of tasks and the number of site are not equal in general, and a software development task is usually be solved through a collaboration among several sites.

Therefore, the task assignment problem in global software development is defined as follows: assign n tasks to m development sites with the least total cost ($m>n$), if task i is assigned to site j with a non-negative integer cost c_{ij}. This problem is a special case of the linear programming problem, defined as follows:

$$\text{Min } S=\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{6}$$

$$\text{Subject to } \sum_{i=1}^{m} x_{ij} = 1 \text{ (j=1, 2, ..., n)}, \tag{7}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \text{ (i=1, 2, ..., m)}, \tag{8}$$

$$x_{ij}=1 \text{ or } 0. \tag{9}$$

For this case, we need to find an optimal assignment given a $n\times m$ cost matrix.

Theorem 2: If a $n\times m$ matrix is transformed into two $n\times n$ matrix by adding ($2n$-m) virtual sites with the zero entries, then an optimal assignment for the resulting cost matrixes is also an optimal assignment for the original cost matrix.

Proof: For the resulting cost matrix C', the problem is defined as follows:

$$\text{Min } S'' = \sum_{i=1}^{n}\sum_{j=1}^{2n} c'_{ij} \times x'_{ij} \tag{10}$$

$$\text{Subject to } \sum_{i=1}^{n} x'_{ij} = 1 \ (j = 1,2,\dots,2n) \tag{11}$$

$$\sum_{j=1}^{n} x'_{ij} = 1 \ (i = 1,2,\dots,n) \tag{12}$$

$$\sum_{j=n+1}^{2n} x'_{ij} = 1 \ (i = 1,2,\dots,n) \tag{13}$$

$$x'_{ij} = 1 \ or \ 0 \ \ (\text{i} = 1,2,\dots,\text{n};\text{j} = 1,2,\dots,2\text{n}) \tag{14}$$

$$\begin{aligned} \text{Min } S'' &= \sum_{i=1}^{n}\sum_{j=1}^{2n} c'_{ij} \times x'_{ij} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{m} c'_{ij}\, x'_{ij} + \sum_{i=1}^{n}\sum_{j=m+1}^{2n} c'_{ij}\, x'_{ij} \end{aligned}$$

Because $c'_{ij} = 0 \ \ (i = 1,2,\dots,n; j = m+1, m+2, \dots, 2n)$

$$\text{Then Min } S'' = \sum_{i=1}^{n}\sum_{j=1}^{2n} c'_{ij} \times x'_{ij} + 0$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{2n} c'_{ij} \times x'_{ij}$$

$$= \text{Min } S'$$

B. The RHA method

The following algorithm applies the above mathematical model and theorem 2 to a given $n \times m$ cost matrix to find an optimal assignment. Algorithm 1 presents the pseudo-code of our proposed RHA method.

Algorithm 1. RHA approach

Input: $n \times m$ cost matrix

Output: optimal assignment result

1. Subtract the smallest entry in each row from all the entries of its row;
2. Subtract the smallest entry in each column from all the entries of its column;
3. Transform $n \times m$ matrix into two $n \times n$ matrix by adding (2n-m) virtual sites with the zero entries;
4. Performs the original Hungarian algorithm on the two $n \times n$ cost matrix**;**
5. Removes the (2n-m) virtual development sites ;
6. **return** optimal assignment result;

IV. EXAMPLE

In this section, we illustrate the execution of RHA by an example.

Example 1: A software development project consists of four different tasks: Task 1, Task 2, Task 3 and Task 4. There are six development sites: S1, S2, S3, S4, S5, and S6. They each demand different pay for various tasks. The problem is to find the lowest-cost way to assign the tasks. The 6×8 cost matrix of this problem is shown as follows.

$$E_{ij}=\begin{bmatrix} 9.4 & 4.8 & 5 & 8.2 & 12 & 7.4 \\ 9.6 & 10 & 6.4 & 8.8 & 8.8 & 9 \\ 6.6 & 11 & 8 & 7.6 & 7.6 & 6.6 \\ 8.2 & 8.4 & 5 & 8.2 & 6.4 & 8.6 \end{bmatrix}$$

Step 1. Subtract 4.8 from Row 1, 6.4 from Row 2, 6.6 from Row 3, and 5 from Row 4. The resulting matrix is as follows.

$$\begin{bmatrix} 4.6 & 0 & 0.2 & 3.4 & 7.2 & 2.6 \\ 3.2 & 3.6 & 0 & 2.4 & 2.4 & 2.6 \\ 0 & 4.4 & 1.4 & 1 & 3.4 & 0 \\ 3.2 & 3.4 & 0 & 3.2 & 1.4 & 3.6 \end{bmatrix}$$

Step 2. Subtract 1 from Column 4, and 1.4 from Column 5. The resulting matrix is as follows.

$$\begin{bmatrix} 4.6 & 0 & 0.2 & 2.4 & 5.8 & 2.6 \\ 3.2 & 3.6 & 0 & 1.4 & 1 & 2.6 \\ 0 & 4.4 & 1.4 & 0 & 2 & 0 \\ 3.2 & 3.4 & 0 & 2.2 & 0 & 3.6 \end{bmatrix}$$

Step 3. Transform 6×4 matrix into two 4×4 matrix by adding 2 virtual sites with the zero entries. The resulting matrixes are as follows.

$$\text{The first matrix:} \begin{bmatrix} 4.6 & 0 & 0.2 & 2.4 \\ 3.2 & 3.6 & 0 & 1.4 \\ 0 & 4.4 & 1.4 & 0 \\ 3.2 & 3.4 & 0 & 2.2 \end{bmatrix}$$

$$\text{The second matrix:} \begin{bmatrix} 5.8 & 2.6 & 0 & 0 \\ 1 & 2.6 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 3.6 & 0 & 0 \end{bmatrix}$$

Step 4. Performs the original Hungarian algorithm on the two 4×4 cost matrix. The assignment result of the first cost matrix is as follows:

Task1--->S2

Task2--->S4

Task3--->S1

Task4--->S3

The assignment result of the second cost matrix is as follows:

Task1--->S7

Task2--->S8

Task3--->S6

Task4--->S5

Step 5.Removes the 2 virtual development sites and output the final assignment result:

Task1--->S2

Task2--->S4

Task3--->S1,S6

Task4--->S3,S5

V. EXPERIMENTS

In this section, we evaluate our proposed reinforced Hungarian algorithm (RHA) for task assignment empirically. We first introduce the performance measures. Then, in order to investigate the performance of RHA, we perform some empirical experiments.

A. *Performance measures*

In the experiment, we employ one commonly used performance measures, i.e, successful allocation rate. It is defined and summarized as follows.

- Successful allocation rate (SAR) is the measure of tasks that are successfully allocated to development sites. The SAR=M_s/M, where M_s is the number of cost matrixes that are successfully allocated and M is the number of all cost matrixes. The higher the value of successful allocation rate, the more effective the algorithm is.

B. *Experimental results*

In this experiment, we analyze the effectiveness of RHA with different number of development sites. Here we fix the number of tasks to 4 and evaluate the performance of the evaluated algorithms by increasing the number of development sites from 6 to 12 with an increment of 2. Therefore, we generated four sets of matrix data, which contains 100 *4×6* matrixes, 100 *4×8* matrixes, 100 *4×10* matrixes and 100 *4×12* matrixes, respectively.

Result Analysis: As we can see in Fig.1, the successful allocation rate of RHA is always 1. The successful allocation rate of the expansion matrix method goes up when the number of development sites increases. It shows that RHA can effectively allocate tasks to development sites with the lowest cost when the number of development sites changes.

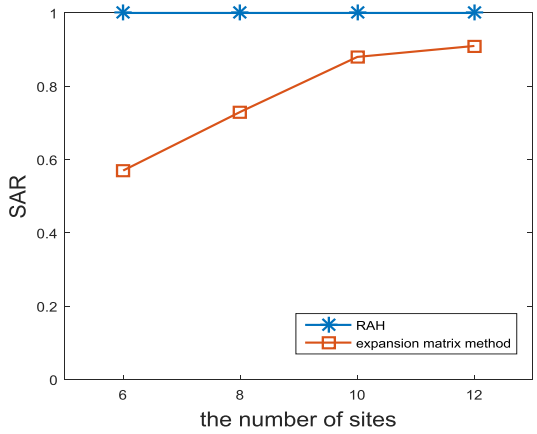


Fig. 1. SAR performances with different number of development sites

Experiment 2: In this experiment, we analyze the effectiveness of RHA with different number of tasks. Here we fix the number of development sites to 4 and evaluate the performance of the evaluated algorithms by increasing the number of tasks from 4 to 10 with an increment of 2. Therefore, we generated four sets of matrix data, which contains 100 *4×4* matrixes, 100 *4×6* matrixes, 100 *4×8* matrixes and 100 *4×10* matrixes, respectively.

Result Analysis: As we can see in Fig.2, the successful allocation rate of RHA is always 1. The successful allocation rate of the expansion matrix method decreases when the number of tasks increases It shows that RHA can effectively allocate tasks to development sites with the lowest cost when the number of tasks changes.

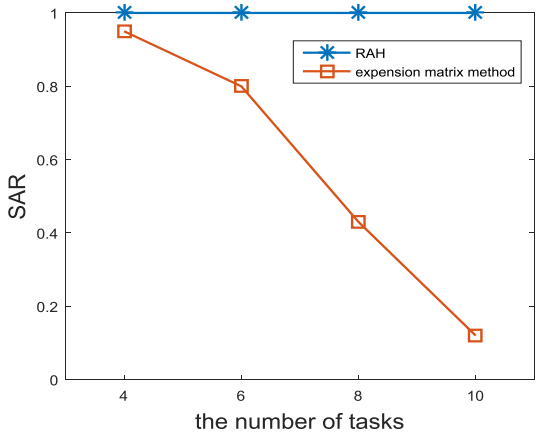


Fig. 2. SAR performances with different number of tasks

VI. Related Work

In this section, we briefly review the existing task assignment methods [13-26] in global software distribution.

Geographically distributed development creates new questions about how to coordinate muhi-site work. Grinter proposes four methods product development organizations to coordinate muhi-site work. He finds that no matter what model is used, it is difficulty to acquire expertise and the distribution of project mass may be unequal [13-14].

Lamersdorf et.al think that distributed development is often driven by some factors which are not distinguished, such as risks, workforce capabilities, the innovation potential of different regions, and cultural factors. They discuss about these factors and find a lack of empirically-based multi-criteria distribution models [15]. A qualitative study aiming to identify and understand the criteria used in practice was conducted by Lamersdorf, et al. The results show that the sourcing strategy and the type of software to be developed have a significant effect on the applied criteria [16]. The criteria and causal relationships were identified in a literature study and refined in a qualitative interview study. Lamersdorf et.al introduce a model which aims at improving management processes in globally distributed projects by giving decision support for task allocation that systematically regards multiple criteria [18]. A planning tool to identify task assignment based on multiple criteria and weighted project goals was developed by Lamersdorf and Munch. The model is called TAMRI (Task Allocation based on Multiple cRIteria) and its implementation is based on Bayesian networks. The application of the tool requires a large amount of knowledge on casual relationships in distributed development. [17].

Though these process models have helped companies to achieve global standards, some social aspects are not considered. Amrit proposes a methodology to test a hypothesis based on how social networks can be used to improve coordination in software industry [21]. Setamanit describes GSD-a hybrid computer simulation model of the software development process in order to identify the practices of work distribution and try to classify criteria [22]. Marques presents a

domain ontology to represent concepts related to task allocation in distributed teams in his paper. This method deals with the lack of standardized vocabulary and achieve knowledge acquisition and sharing [23].

VII. Conclusion and Future Work

In this paper, we address the issue of how to find the lowest-cost way to allocate tasks among development sites. We propose a reinforced Hungarian algorithm (RHA) for task assignment in global software development. RHA consist of two stages. In the first stage, the $n \times m$ matrix are transformed into two $n \times n$ matrix by adding virtual development sites. In the second stage, the two $n \times n$ matrix are solved by the original Hungarian algorithm. Finally, the virtual tasks are removed to get the optimal assignment results. We conduct experiments on the synthetic datasets to evaluate the performance of the proposed algorithm. The experimental results indicate that the proposed algorithm can effectively allocate tasks to development sites with the lowest cost.

In the future, we would like to validate the generalization ability of our method on available real-world datasets. In addition, we plan to investigate the task assignment method based on social data [24-25].

Acknowledgment

This work is partly supported by the grants of National Natural Science Foundation of China (No.61572374, No.U163620068, No.U1135005) and the Academic Team Building Plan from Wuhan University and National Science Foundation (NSF) (No. DGE-1522883).

References

[1] Taylor P S, Greer D, Sage P, et al. Do agile GSD experience reports help the practitioner?[C]//Proceedings of the 2006 international workshop on Global software development for the practitioner. ACM, 2006: 87-93.

[2] DeLone W, Espinosa J A, Lee G, et al. Bridging global boundaries for IS project success[C]//System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on. IEEE, 2005: 48b-48b.

[3] Xu Z, Liu Y, Mei L, et al. Semantic based representing and organizing surveillance big data using video structural description technology[J]. Journal of Systems and Software, 2015, 102: 217-225.

[4] Liu J, Yu X, Xu Z, et al. A cloud‐based taxi trace mining framework for smart city[J]. Software: Practice and Experience, 2016.

[5] Liu Z, Wei C, Ma Y, et al. UCOR: an unequally clustering-based hierarchical opportunistic routing protocol for WSNs[C]//International Conference on Wireless Algorithms, Systems, and Applications. Springer Berlin Heidelberg, 2013: 175-185.

[6] Liu Z, Niu X, Lin X, et al. A Task-Centric Cooperative Sensing Scheme for Mobile Crowdsourcing Systems[J]. Sensors, 2016, 16(5): 746.

[7] Huda N, Nahar N, Tepandi J, et al. Key barriers for global software product development organizations[C]//Management of Engineering & Technology, 2009. PICMET 2009. Portland International Conference on. IEEE, 2009: 1081-1087.

[8] Šmite D, Borzovs J. Managing uncertainty in globally distributed software development projects[J]. University of Latvia, Computer Science and Information Technologies, 2008, 733: 9-23.

[9] Wickramaarachchi D, Lai R. A method for work distribution in global software development[C]//Advance Computing Conference (IACC), 2013 IEEE 3rd International. IEEE, 2013: 1443-1448.

[10] Bass M, Paulish D. Global software development process research at Siemens[C]//Third International Workshop on Global Software Development. 2004: 8-11.

[11] Jonker R, Volgenant T. Improving the Hungarian assignment algorithm[J]. Operations Research Letters, 1986, 5(4): 171-175.

[12] Yexin S, Mianyun C, Shuhong Z. An efficient algorithm for solving two multi-object generalized assignment problems and its application[J]. JOURNAL-HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY CHINESE EDITION, 2001, 29(1): 70-72.

[13] Grinter R E, Herbsleb J D, Perry D E. The geography of coordination: Dealing with distance in R&D work[C]//Proceedings of the international ACM SIGGROUP conference on Supporting group work. ACM, 1999: 306-315.

[14] Edwards H K, Kim J H, Park S, et al. Global software development: Project decomposition and task allocation[C]//International Conference on Business and Information. 2008.

[15] Lamersdorf A, Münch J, Rombach D. Towards a multi-criteria development distribution model: An analysis of existing task distribution approaches[C]//Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on. IEEE, 2008: 109-118.

[16] Lamersdorf A, Munch J, Rombach D. A survey on the state of the practice in distributed software development: Criteria for task allocation[C]//Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on. IEEE, 2009: 41-50.

[17] Lamersdorf A, Munch J. TAMRI: a tool for supporting task distribution in global software development projects[C]//Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on. IEEE, 2009: 322-327.

[18] Lamersdorf A, Münch J, Rombach D. A decision model for supporting task allocation processes in global software development[C]//International Conference on Product-Focused Software Process Improvement. Springer Berlin Heidelberg, 2009: 332-346.

[19] Lamersdorf A. Empirically-Based Decision Support for Task Allocation in Global Software Development[C]//ICGSE. 2009: 281-284.

[20] Damian D, Moitra D. Guest editors' introduction: Global software development: How far have we come?[J]. IEEE software, 2006, 23(5): 17-19.

[21] Amrit C. Coordination in software development: the problem of task allocation[C]//ACM SIGSOFT Software Engineering Notes. ACM, 2005, 30(4): 1-7.

[22] Setamanit S, Wakeland W, Raffo D. Using simulation to evaluate global software development task allocation strategies[J]. Software Process: Improvement and Practice, 2007, 12(5): 491-503.

[23] Marques A B, Carvalho J R, Rodrigues R, et al. An ontology for task allocation to teams in distributed software development[C]//Global Software Engineering (ICGSE), 2013 IEEE 8th International Conference on. IEEE, 2013: 21-30.

[24] Xu Z, Mei L, Lu Z, et al. Multi-modal Description of Public Security Events using Surveillance and Social Data[J]. IEEE Transactions on Big Data, 2017.

[25] Xu Z, Zhang H, Hu C, et al. Building knowledge base of urban emergency events based on crowdsourcing of social media[J]. Concurrency and Computation: Practice and Experience, 2016.

A Data Filtering Method Based on Agglomerative Clustering

Xiao Yu
State Key Lab. of Software Engineering,
Computer School, Wuhan University,
Wuhan, China
xiaoyu_whu@yahoo.com

Jiansheng Zhang
School of Computer Science and Information Engineering,
HuBei University,
Wuhan, China
jianjian_zhang@qq.com

Peipei Zhou
School of Computer Science and Information Engineering,
HuBei University,
Wuhan, China

Jin Liu*
State Key Lab. of Software Engineering,
Computer School, Wuhan University,
Wuhan, China
Corresponding author email: jinliu@whu.edu.cn

***Abstract*—Cross-company defect prediction (CCDP) is a practical way that trains a prediction model by exploiting one or multiple projects of a source company and then applies the model to target company. Unfortunately, larger irrelevant cross-company (CC) data usually makes it difficult to build a cross-company defect prediction model with high performance. To address such issues, this paper proposes a data filtering method based on Agglomerative Clustering (DFAC) for cross-company defect prediction. First, DFAC combines within-company instances and cross-company instances and uses Agglomerative clustering algorithms to group these instances. Second, DFAC selects sub-clusters which consist at least one WC instance, and collects the CC instances in the selected sub-clusters into a new CC data. Compared with existing data filter methods, the experimental results on 15 public PROMISE datasets show that DFAC increases PD value, reduces PF value and achieves higher G-measure and AUC values.**

Keywords—software defect prediction;cross-company defect prediction;data filter; Agglomerative clustering

I. INTRODUCTION

Software defect prediction is one of the most important software quality assurance techniques. Based on the investigation of historical metrics [1-2], 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 [3-5]. With the advent of big data era and the development of machine learning techniques [6], many machine learning algorithms are applied to solve the practical problems in life [7-9].

DOI reference number: 10.18293/SEKE2017-043

In the same way, many efficient software defect prediction methods using statistical methods or machine learning techniques have been proposed [10-21], but they are usually confined to predicting a given software module being faulty or non-faulty by means of some binary classification techniques. WPDP works well if sufficient data is available to train a defect prediction model. However, it is difficult for a new project to perform WPDP if there is limited historical data. Cross-company defect prediction (CCDP) is a practical approach to solve the problem. It trains a prediction model by exploiting one or multiple projects of a source company and then applies the model to target company [22].

In recent years, most existing CCDP approaches have been proposed. Unfortunately, larger irrelevant cross-company (CC) data usually makes it difficult to build a cross-company defect prediction model with high performance. Therefore, how to weaken the impact of irrelevant CC data to improve the performance of CCDP is a big challenge. The ability to transfer knowledge from a source company to a target company depends on how they are related. The stronger the relationship, the more usable will be CC data. The performance of CCDP is generally poor because of larger irrelevant CC data. Previous work [23] found that using raw CC data directly would increase false alarm rates due to irrelevant instance in CC data, so several data filtering works should be done before building the prediction model. For example, Turhan et al. [23] and Peters et al. [24] proposed the NN filter and the Perters filter to select the CC instances which are mostly similar to WC data as the training dataset.

Considering such challenge, this paper proposes a data filtering method based on Agglomerative clustering (DFAC) for cross-company defect prediction. First, DFAC combines within-company instances and cross-company instances and

uses Agglomerative clustering algorithms to group these instances. Second, DFAC selects sub-clusters which consist at least one WC instance, and collects the CC instances in the selected sub-clusters into a new CC data. We evaluate our proposed method, DFAC, on 15 publicly available project datasets. Experimental results show that DFAC can effectively filter out the irrelevant CC instances to improve the overall prediction performance. On most of project datasets under evaluation, DFAC performs the best G-measure values.

The remainder of this paper is organized as follows. Section II presents the related work. Section III describes our proposed DFAC method. Section IV demonstrates the experimental results. Section V discusses the potential threats to validity. Finally, Section VI addresses the conclusion and points out the future work.

II. Related Work

In this section, we first review the existing defect prediction methods. Then, we briefly review the cross-company defect prediction or cross-project defect prediction.

A. Defect prediction

Many researchers have proposed various models for predicting the module being faulty or non-faulty in terms of within-project defect prediction (WPDP).

Support vector machine [10], neural networks [11-13], and decision trees [14-15] paved the way for classification-based methods in the flied of defect prediction. These methods used software metrics to properly predict whether a module is defect-prone or not. However, feature irrelevance and imbalanced nature of the defect datasets degraded the prediction performance. Therefore, some feature selection methods [16-17] and class imbalance learning methods [18] have been proposed to cope with feature irrelevance and class imbalance problem for software defect prediction. For example, Wang et al. [17] leveraged Deep Belief Network to automatically learn semantic features from source code.

In the process of defect prediction, misclassify different software defect classes can be divided into two types, namely, "Type I" and "Type II". "Type I" misclassification cost and "Type II" misclassification cost are different. Therefore, some cost-sensitive learning methods [19] have been proposed to address this issue by generating a classification model with minimum misclassification cost. In addition, several methods [20-21] based on ensemble learning have been proposed to address the defect prediction problem. However, these methods are confined to predicting a given software module being faulty or non-faulty.

B. Cross-company defect prediction

In order to solve the problem that the new companies have too limited historical data to perform WCDP well, the cross-project and cross-company defect prediction appeared. Zimmermann et al. [25] studied CCDP models on 12 real-world applications datasets. Their results indicated that CCDP is still a serious challenge because of the different distribution between the training project data and the target project data. In order to narrow the distribution gap, there are three mainstream ways.

The first one is to apply the data filtering method to find the best suitable training data (e.g., [23, 24, 26]). For example, Turhan et al. [23] proposed a nearest neighbor (NN) filter to select cross-company data. Peters et al. [24] introduced the Peters filter to select training data via the structure of other projects. They compared the filter with two other approaches for quality prediction to assess the performance of the Peters filter, and found that 1) WCDP are weak for small data sets; 2) the Peters filter + CCDP builds better and more useful predictors.

The second mainstream way is to design effective defect predictor based on transfer learning techniques (e.g., [22, 27, 28]). For instance, Ma et al. [27] proposed a novel algorithm called Transfer Naive Bayes (TNB) to transfer cross-company data information into the weights of the training data and then build the predictor based on re-weighted CC data. The results indicated that TNB is more accurate in terms of AUC, within less runtime than the state of the art methods and can effectively achieve the CCDP task. Chen et al. [28] proposed double transfer boosting (DTB) model. Another challenge in CCDP is that the set of metrics between the source company data and target company data is usually heterogeneous. The heterogeneous CCDP (HCCDP) task is that the source and target company data is heterogeneous. Jing et al. [22] provided an effective solution for HCCDP. They proposed a unified metric representation (UMR) for the data of source and target companies and introduced canonical correlation analysis (CCA), an effective transfer learning method, into CCDP to make the data distributions of source and target companies similar. Results showed that their approach significantly outperforms state-of-the-art CCDP methods for HCCDP with partially different metrics and for HCCDP with totally different metrics, their approach is also effective.

The third mainstream way is to apply unsupervised classifier that does not require any training data to perform CCDP (e.g., [29-30]), therefore the distribution gap between the training project data and the target project data is no longer an issue. For instance, Zhang et.al [30] proposed to apply a connectivity-based unsupervised classifier that is based on Agglomerative clustering to perform CPDP.

III. Methodology

A. DFAC

Previous work [23] found that using raw CC data directly would increase false alarm rates due to irrelevant instances in CC data, so several data filtering works should be done before building the prediction model. The main goal of data filter is to select the most valuable training data for the CCDP model by filtering out irrelevant instances in CC data. In this paper, we propose a Data Filtering method based on Agglomerative Clustering algorithm (DFAC).

DFAC consists of two stages. In the first stage, DFAC combines within-company instances and cross-company instances and uses Agglomerative clustering algorithms to

group these instances. The main goal of agglomerative clustering is to partition WC instances and CC instances into k clusters such that instances in the same cluster are similar and instances in different clusters are dissimilar to each other.

Agglomerative clustering is an iterative process, which merges current clusters continuously. It is possible that a current cluster only contains one instance, e.g., each instance is treated as one cluster at the beginning of the iteration. In agglomerative clustering, instances are merged into clusters according to the distances between current clusters. We employ the average linkage method to define the distance of two current clusters. The average linkage between two clusters is defined as the average of the distance between any instance pair between two clusters. Suppose that U_a and U_b are two current clusters during the clustering process. The distance of the two clusters $D_{a,b}$ can be calculated by the following formula:

$$D_{a,b} = \frac{1}{m_a m_b} \sum_{x_i \in U_a, x_j \in U_b} d_{i,j} \quad (1)$$

where $m_a$and m_b are the number of instances inside clusters U_a and U_b and $d_{i,j}$ is the distance between two instances x_i and x_j. We define the distance $d_{i,j}$ of two instances x_i and x_j with Euclidean distance.

DFAC assumes that CC instances which are in the same cluster as WC instances are the most valuable instances in CC data. Therefore, in the second stage, DFAC selects sub-clusters which consist at least one WC instance, and collects the CC instances in the selected sub-clusters into a new CC data.

B. Example

Fig.1 shows the resulting clusters of a set of instances using the Agglomerative clustering algorithm, where "○" represents the CC instance and "△"represents the WC instance.

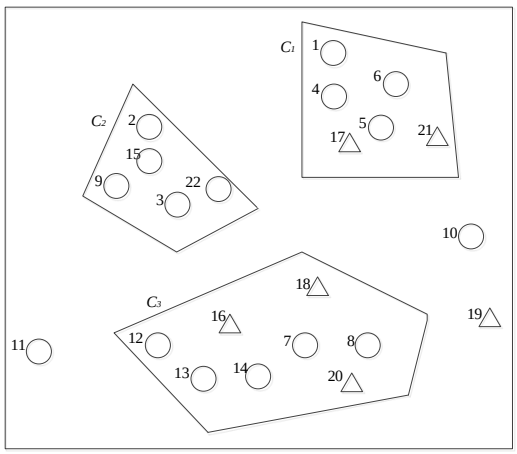


Fig. 1. Resulting clusters of a set of instances using Agglomerative clustering algorithm

These instances are partitioned into three clusters by using the Agglomerative clustering algorithm, namely, C_1={x_1, x_4, x_5, x_6, x_{17}, x_{21}}, C_2={x_2, x_3, x_9, x_{15}, x_{22}}, and C_3={x_7, x_8, x_{12}, x_{13}, x_{14}, x_{16}, x_{18}, x_{20}}. Take the cluster C_1 for example, since the CC instances x_1, x_4, x_5 and x_6 are in the same cluster as the WC instance x_{17} and x_{21}, these CC instances are selected to form the final CC training data. Take the cluster C_2 for example, since the cluster do not consist any WC instance, the CC instances in this cluster are discarded. The CC instances in the cluster C_3 are selected in the same manner. Therefore, the final CC training instances consist of x_1, x_4, x_5, x_6, x_7, x_8, x_{12}, x_{13} and x_{14}.

IV. Experiments

In this section, we evaluate our DFAC method to perform CCDP empirically. We first introduce the experiment dataset, the performance measures and the experimental procedure.

A. Data set

In this experiment, we employ 15 available and commonly used datasets which can be obtained from PROMISE [31]. The 15 datasets have the same 20 attributes, so we can apply all attribute information directly. Table I tabulates the details about the datasets, and Table II shows a more detailed description of the 20 independent code attributes.

TABLE I. Details of Experiment Dataset

Project	***Examples***	***%Defective***	***Description***
ant	125	16	Open-source
arc	234	11.5	Academic
camel	339	3.8	Open-source
elearn	64	7.8	Academic
jedit	272	33.1	Open-source
log4j	135	25.2	Open-source
lucene	195	46.7	Open-source
poi	237	59.5	Open-source
prop	660	10	Proprietary
redaktor	176	15.3	Academic
synapse	157	10.2	Open-source
system	65	13.8	Open-source
tomcat	858	9	Open-source
xalan	723	15.2	Open-source
xerces	162	47.5	Open-source

TABLE II. Code Attributes of the Datasets

No.	***Attribute***	***Description***
1	*wmc*	Weighted methods per class
2	*dit*	Depth of inheritance tree
3	*noc*	Number of children
4	*cbo*	Coupling between object classes
5	*rfc*	Response for a class
6	*lcom*	Lack of cohesion in methods
7	*ca*	Afferent couplings
8	*ce*	Efferent couplings
9	*npm*	Number of public methods
10	*lcom3*	Lack of cohesion in methods
11	*loc*	Lines of code
12	*dam*	Data access metric
13	*moa*	Measure of aggregation
14	*mfa*	Measure of functional abstraction
15	*cam*	Cohesion among methods of class
16	*ic*	Inheritance coupling
17	*cbm*	Coupling between methods
18	*amc*	Average method complexity
19	*max_cc*	Maximum McCabe's cyclomatic complexity
20	*avg_cc*	Average McCabe's cyclomatic complexity

B. Performance measures

In the experiment, we employ three commonly used performance measures including *pd*, *pf* and *g-measure*. They are defined in Table 2 and summarized as follows.

TABLE III. PERFORMANCE MEASURES

		Actual	
		yes	***no***
Predicted	***yes***	***TP***	***FP***
	no	***FN***	***TN***
pd	$\frac{TP}{TP+FN}$		
pf	$\frac{FP}{FP+TN}$		
G-measure	$\frac{2*pd*(1-pf)}{pd+(1-pf)}$		

- Probability of detection or pd is the measure of defective modules that are correctly predicted within the defective class. The higher the pd, the fewer the false negative results.

- Probability of false alarm or pf is the measure of non-defective modules that are incorrectly predicted within the non-defective class. Unlike pd, the lower the pf value, the better the results.

- G-measure is a trade-off measure that balances the performance between pd and pf. A good prediction model should have high pd and low pf, and thus leading to a high g-measure.

C. Experimental Procedure

In order to confirm whether DFAC can perform better than other data filtering methods, we compare DFAC with four state-of-the-art CCDP approaches. More details are provided below:

- **NN filter** [23] is based on the widely used classification method K-Nearest Neighbors (KNN) algorithm to filter irrelevant CC data. It can find out the most similar $K \times N$ instances from CC data while N is the number of instances in WC data and K is the parameter of the KNN method. In our experiment, we choose K as 10.

- **DBSCAN filter** [26] is based on the DBSCAN algorithm. DBSCAN defines the high density with two parameters: the distance which determines whether two records are close to and the number of records which determines whether a core sample is in a dense area. In our experiment, we choose the two parameters as 10 and 10, respectively.

In every experiment, one dataset is selected as WC data and the rest are regarded as CC data to conduct the experiment. The CC data is considered as basic training data which will be adjusted in every experiment. All data filtering methods are done on CC data. Then processed CC data are used to build the CCDP model. Finally, the resulting model is evaluated on the WC data. The procedure will be repeated 30 times in every experiment to avoid sample bias. Then, the mean values of performance are calculated. In this experiment, we choose Naive Bayes (NB) [32] as the CCDP model due to its effectiveness in defects prediction.

D. Experiment results

The comparison results of 15 projects with NB classifier on pd, pf and G-measure are summarized in Table IV. Table IV shows that DFAC performs better average pd and pf values than all the other data filtering methods. It shows that the NN filter often achieves good pd but the worst pf so that it usually ends up with low g-measure value. The performance of DBSCAN filter seems sometimes have lower pf value than the NN filter but it has lower pd value. It's very clear that DFAC has lower pf value and higher pd value than the other two data filtering methods. In the aspect of pd value, the DFAC approach increases the pd value to an extent on most data sets. The pd values of 8 projects are better than others. On most tests, DFAC achieves higher G-measure than the other data filtering methods. In total, DFAC has acceptable pf value and can obtain better pd values in most experiments we conducted, and it almost always achieve the higher G-measure value than other data filtering methods. In other words, the DFAC approach outperforms other methods, therefore it can be an effective data filtering methods.

Fig. 2 shows the box-plots of G-measure values, for NN filter, DBSCAN filter and DFAC, with NB classifier on 15 projects. It seems that NN filter approach always has low G-measure comparing to DBSACN filter and DFAC. Although DFAC has the same minimum with DBSCAN filter, the median values of both metrics of DFAC are higher than those of NN filter and DBSCAN filter. Meanwhile, the maximum by DFAC is higher than that by DBSCAN filter, but a litter lower than that by NN filter.

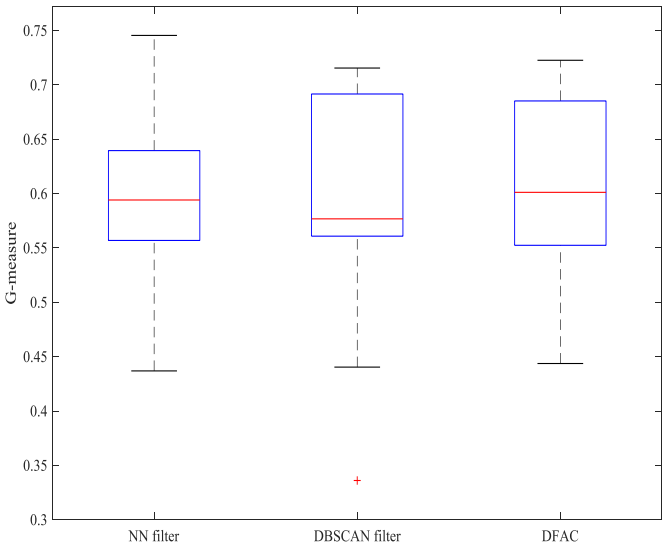


Fig. 2. Box-plots for G-measure on 15 projects.

In total, DFAC has acceptable pd value and can obtain better pf values in most experiments we conducted, and it almost always achieve the higher G-measure than other data filtering methods. In other words, DFAC outperforms other data filtering methods.

TABLE IV. PD, PF AND G-MEASURE VALUES ON 15 PROJECTS

No.	Test data	NN filter			DBSCAN filter			DFAC		
		PD	PF	G	PD	PF	G	PD	PF	G
1	ant	**0.829**	0.537	0.594	0.768	**0.539**	0.576	0.813	0.485	**0.631**
2	arc	0.829	0.593	0.546	0.824	**0.594**	0.544	**0.838**	0.592	**0.549**
3	camel	0.889	**0.879**	0.213	0.875	0.792	0.336	**0.902**	0.704	**0.446**
4	elearn	0.842	**0.705**	0.437	0.860	0.704	0.440	**0.877**	0.703	**0.444**
5	jedit	0.779	0.337	0.716	0.775	**0.345**	0.710	**0.783**	0.329	**0.723**
6	log4j	**0.860**	0.342	**0.746**	0.835	**0.374**	0.716	0.843	0.372	0.720
7	lucene	**0.657**	0.374	**0.641**	0.646	**0.383**	0.631	0.651	0.378	0.636
8	poi	**0.545**	0.338	**0.598**	0.521	**0.354**	0.577	0.531	0.352	0.584
9	prop-6	**0.803**	**0.531**	0.592	0.801	0.370	**0.705**	0.796	0.517	0.601
10	redactor	0.747	**0.372**	0.682	**0.759**	0.338	**0.707**	**0.759**	0.338	**0.707**
11	synapse	0.759	**0.508**	0.597	0.773	0.438	0.651	**0.787**	0.367	**0.702**
12	system	**0.828**	**0.485**	**0.635**	**0.828**	**0.485**	**0.635**	**0.828**	**0.485**	**0.635**
13	tomcat	0.854	0.591	0.553	0.850	0.577	**0.565**	**0.858**	**0.604**	0.542
14	xalan	**0.820**	0.565	**0.568**	0.817	**0.574**	0.560	0.818	0.565	**0.568**
15	xerces	**0.579**	0.425	**0.577**	0.566	**0.439**	0.563	0.566	**0.439**	0.563
	Average	0.775	0.505	0.580	0.767	0.487	0.594	**0.777**	**0.482**	**0.603**

V. THREATS TO VALIDITY

In this section, we discuss several validity threats that may have an impact on the results of our studies.

External validity. Threats to external validity occur when the results of our experiments cannot be generalized. As a preliminary result, we performed our experiments on the 15 datasets to explore the generality of our method. Although these datasets have been widely used in many software defect prediction studies, we still cannot claim that our method can be generalized to other datasets. Nevertheless, this work provides a detail experimental description, including parameter settings (default parameter settings specified by *sklearn*), thus other researchers can easily replicate our method on new datasets.

Internal validity. In our study, we repeat 30 times to avoid sample bias, and calculate average results to verify the performance of all test methods. We compared our method with NN filter and DBSCAN filter. To avoid the potential faults during the implementation process of the experiment, we implement these models based on the python machine learning library *sklearn*.

Construct validity. Threats to construct validity focus on the bias of the measures used to evaluate the performance of CCDP. In our experiments, we mainly use pd, pf, G-measure to measure the effectiveness of the four approaches. Nonetheless, other evaluation measures such as AUC measure can also be considered.

VI. CONCLUSION AND FUTURE WORK

In this paper, we propose a data filtering method based on Agglomerative clustering for CCDP. The method consists of two stages. In the first stage, DFAC combines WC instances and CC instances and uses Agglomerative clustering algorithms to group these instance. In the second stage, DFAC selects sub-clusters which consist at least one WC instance, and collects the CC instances in the selected sub-clusters into a new CC data. We conduct experiments on the 15 datasets to evaluate the performance of the proposed DFAC method. The experimental results indicate that the proposed method can effectively filter out and weaken the impact of irrelevant data to improve the performance of CCDP. The proposed DFAC method is an effective data filtering method for CCDP.

In the future, we would like to employ more project datasets to validate the generalization of our proposed method. In addition, we plan to apply our method to a real-life application [33-34].

ACKNOWLEDGMENT

This work is partly supported by the grants of National Natural Science Foundation of China (No.61572374, No.U163620068, No.U1135005) and the Academic Team Building Plan from Wuhan University and National Science Foundation (NSF) (No. DGE-1522883).

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] K. Gao, T.M. Khoshgoftaar, H. Wang, et al, “Choosing software metrics for defect prediction: an investigation on feature selection techniques ,” Software Practice & Experience, 2011,41(5):579-606.

[3] M. Shepperd, D. Bowes, T. Hall, “Researcher Bias: The Use of Machine Learning in Software Defect Prediction,” IEEE Transactions on Software Engineering, 2014,40(6):603-616.

[4] Q. Song, Z. Jia, M. Shepperd, et al, “A general software defect proneness prediction framework,” Software Engineering, IEEE Transactions on, 2011, 37(3): 356-370.

[5] X. Yang, K. Tang, X. Yao, “A Learning-to-Rank Approach to Software Defect Prediction,” IEEE Transactions on Reliability, 2015,64(1): 234-246.

[6] Xu Z, Liu Y, Mei L, et al. “Semantic based representing and organizing surveillance big data using video structural description technology,” Journal of Systems and Software, 2015, 102: 217-225.

[7] Xu Z, Zhang S, Choo K K, et al. “Hierarchy-cutting model based association semantic for analyzing domain topic on the web” IEEE Transactions on Industrial Informatics, 2017.

[8] Xu Z, Mei L, Lu Z, et al. “Multi-modal Description of Public Security Events using Surveillance and Social Data” IEEE Transactions on Big Data, 2017.

[9] Xu Z, Liu Y, Mei L, et al. “The mobile media based emergency management of web events influence in cyber-physical space” Wireless Personal Communications, 2016: 1-14.

[10] Z. Yan, X. Chen, P. Guo, Software defect prediction using fuzzy support vector regression, International Symposium on Neural Networks. Springer Berlin Heidelberg, 2010, pp. 17-24.

[11] Arar, Ömer Faruk, Kürşat Ayan, Software defect prediction using cost-sensitive neural network. Applied Soft Computing 33 (2015) 263-277.

[12] V. Vashisht, M. Lal, G. S. Sureshchandar, et al, A framework for software defect prediction using neural networks. Journal of Software Engineering and Applications, 8.8 (2015) 384.

[13] Erturk, Ezgi, Ebru Akcapinar Sezer, Iterative software fault prediction with a hybrid approach. Applied Soft Computing, 49 (2016) 1020-1033.

[14] J. Wang, B. Shen, Y.Chen, Compressed C4. 5 models for software defect prediction,in: 12th International Conference on Quality Software, 2012, pp. 13-16.

[15] N. Seliya, T. M..Khoshgoftaar, The use of decision trees for cost‐sensitive classification: an empirical study in software quality prediction. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1.5 (2011) 448-459.

[16] Z. Xu, J. Xuan, J. Liu, et al, “MICHAC: Defect Prediction via Feature Selection based on Maximal Information Coefficient with Hierarchical Agglomerative Clustering,” 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER). IEEE, 2016, 1: 370-381.

[17] S. Wang, T. Liu, L. Tan, “Automatically learning semantic features for defect prediction,” Proceedings of the 38th International Conference on Software Engineering. ACM, 2016: 297-308.

[18] S. Wang, X. Yao, “Using class imbalance learning for software defect prediction,” IEEE Transactions on Reliability,2013,62(2):434-443.

[19] X. Y. Jing, S. Ying, Z. W. Zhang, S. S. Wu, and J. Liu, Dictionary learning based software defect prediction, in: Proc. of the 36th International Conference on Software Engineering, 2014, pp. 414-423.

[20] I. H. Laradji, M. Alshayeb, L. Ghouti, Software defect prediction using ensemble learning on selected features, Inform. Softw. Technol. 58 (2015) 388-402.

[21] M. J. Siers, M. Z. Islam, Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem, Information Systems 51 (2015) 62-71.

[22] X. Jing et al, Heterogeneous Cross-Company Defect Prediction by Unified Metric Representation and CCA-Based Transfer Learning, in: Proc. of the 10th Joint Meeting on Foundations of Software Engineering, 2015, pp. 496-507.

[23] B. Turhan, T. Menzies, A.B. Bener, J. Di Stefano, “On the relative value of cross company and within-company data for defect prediction,” Empirical Softw. Eng. 2009,14 (5): 540–578.

[24] Peters F, Menzies T, Marcus A, “Better cross company defect prediction,” In: Proc. of the 10th International Workshop on Mining Software Repositories, San Francisco, CA, 2013, 409-418.

[25] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, B. Murphy, “Cross-project defect prediction: a large scale experiment on data vs. domain vs. process,” in: Proc. of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ACM, 2009, pp. 91–100.

[26] K. Kawata, S. Amasaki, T. Yokogawa, “Improving Relevancy Filter Methods for Cross-Project Defect Prediction,” Applied Computing & Information Technology. Springer International Publishing, 2016: 1-12.

[27] Y. Ma, G. Luo, X. Zeng, A. Chen, “Transfer learning for cross-company software defect prediction,” Inform. Softw. Technol. 2012,54 (3): 248–256.

[28] Chen L, Fang B, Shang Z, et al. “Negative samples reduction in cross-company software defects prediction,” Information and Software Technology, 2015, 62: 67-77.

[29] P. S. Bishnu, V. Bhattacherjee, “Software fault prediction using quad tree-based k-means clustering algorithm,” IEEE Transactions on knowledge and data engineering, 2012, 24(6): 1146-1150.

[30] F. Zhang, Q. Zheng, Y. Zou, et al, “Cross-project defect prediction using a connectivity-based unsupervised classifier,” Proceedings of the 38th International Conference on Software Engineering. ACM, 2016: 309-320.

[31] G. Boetticher, T. Menzies, T. Ostrand, The PROMISE Repository of Empirical Software Engineering Data, 2007 <http://promisedata.org/repository>.

[32] D. D. Lewis, Naive (Bayes) at forty the independence assumption in information retrieval, in: European conference on machine learning, 1998, pp. 4–15.

[33] Liu Z, Wei C, Ma Y, et al. UCOR: an unequally clustering-based hierarchical opportunistic routing protocol for WSNs, International Conference on Wireless Algorithms, Systems, and Applications. Springer Berlin Heidelberg, 2013: 175-185.

[34] Liu Z, Wei C, Ma Y, et al. UCOR: an unequally clustering-based hierarchical opportunistic routing protocol for WSNs, International Conference on Wireless Algorithms, Systems, and Applications. Springer Berlin Heidelberg, 2013: 175-185.

An Ontology-based Approach to Semantic Health Resource Knowledge Base Development for Crisis Preparation Decision Support System

Min Zhu
Institute of Public Safety Research(IPSR), Tsinghua University.
Navy General Hospital of PLA
Beijing, China
zhum14@mails.tsinghua.edu.cn

Jia Qu (co-first author)
Navy General Hospital of PLA
Beijing, China
vzcg6000@126.com

Ruxue Chen (co-first author)
Navy General Hospital of PLA
Beijing, China
13381207330@126.com

Xinzhi Wang
Institute of Public Safety Research(IPSR), Tsinghua University.
Beijing, China
wxz15@mails.tsinghua.edu.cn

Quanyi Huang
Institute of Public Safety Research(IPSR), Tsinghua University.
Beijing, China
qyhuang@tsinghua.edu.cn

Shaobo Zhong* (corresponding author)
Institute of Public Safety Research(IPSR), Tsinghua University.
Beijing, China
zhongshaobo@tsinghua.edu.cn*

***Abstract*—Emergent health resource supply is the key support to the accomplishment of medical rescue mission. It contains huge data of various kinds of health resource to deal with. Soeme researches were directed to solve such problem, including using various optimization models (OM) and decision support system(DSS). However, these methods were not widely used. One major impediment is the lack of standard knowledge representation models to guide the knowledgebase development. Our research presents a new approach to develop a semantic health resource knowledge base (SHRKB), based on a semantic health resource knowledge ontology (SHRKO) using HL7 RIM to represent emergent medical rescue domain knowledge. We also build semantic health resource expression repository (SHRER) and queries encoded in SPARQL format to extend the SHRKO with dynamic decision logic. The method is validated in a case study of the medical assistance mission executed by the hospital ship Peace Ark. The results show that SHRKB could support same decision method come to more effective plan than normal knowledge base. Future, a more efficient DSS for crisis health resource preparation could be established based on the SHRKB.**

Keywords- ontology; semantic; knowledge base; decision support; crisis preparation.

I. INTRODUCTION & RELATED WORK

Emergent health resource supply is the key problem for curing people wounded in the unexpected crisis. Some research showed that the mortality and the disability rate had a linear relationship with the time when starting to treat the wounded.[1,2] So, the emergent health resource should be prepared as quickly as possible to support the curing treatment. However, the preparation is very complex because huge data of various types of health resources should be dealt with and several factors should be taken into concern. Different types of casualty would occur and hundreds types of medicine and medical equipments. There were many researches on the emergency supply management. Major researches were on four aspects, the selecting of location for storing[3,4], the distribution [5,6], the scheduling [7] and the supplement [8]. Optimization models(OM) and decision support systems (DSS) had become increasingly used in these researches to facilitate the management of emergency supply [9,10,11]. OM and DSS could substantially improve the supply effectiveness. However, despite its great potential and a history of successful cases to this day, DSS have not found widely use in medical rescue tasks outside of one or two medical maneuvers. One of the major impediments to its widespread adoption is the lack of standard semantic based knowledge representation models to guide the development of the knowledge base, which is the core component of a DSS for medical rescue tasks.

The ontology, originally defined as "a formal, explicit specification of a shared conceptualization"[12], has been considered as a key technology in knowledge engineering. Medical is a complicated domain. Compared with the data in other domains, there are too many natural language descriptions and terminologies in medical data. As a result, a variety of ontologies have been constructed in medical domain, for example, the Biocaster ontology [13], Unified

10.18293/SEKE2017-076

Medical Language System (UMLS) Semantic Network System(Unified Medical Language System Semantic Network Documentation)[14], etc. On one hand, because of the distributed nature of ontology development, the proliferation of medical ontologies often have a feature of heterogeneity and overlapping domains. On the other hand, little research has been conducted on the rescue medical resource ontologies development. Both would obstacle to knowledge integration in emergent medical rescue domain. The ontologies we constructed in our research also made use of general medical domain ontologies already existing to describe general medical knowledge.

Our research presents a new approach to develop a semantic health resource knowledge base (SHRKB), based on a semantic health resource knowledge ontology (SHRKO) using Health Level Seven International (HL7) reference information model (RIM) to represent emergent medical rescue domain knowledge. We also build a semantic health resource expression repository (SHRER) of rules encoded in Jena rule format to extend the SHRKO with dynamic decision logic. The proposed method is validated in a case study of the overseas humanitarian medical assistance mission, to demonstrate its technical feasibility.

II. TASK DESCRIPTION

The methods and steps used in our research are as follows. First, we set the target domain and acquire the domain knowledge. Second, we use HL7 RIM to present emergent medical rescue domain knowledge. HL7 all message specifications are based on the RIM, a shared information model for integrating entity data and medical domain knowledge[15], which can serve as the basis of unified domain ontology development. We extract the core concepts, properties, attributes and links of the knowledge. Then we use top-down design pattern to develop ontologies. Third, we use web ontology language (OWL) to describe the meta ontology and add ontology instances to form the ontology base. Fourth, we build a semantic health resource expression repository (SHRER) of rules encoded in Jena rule format. Finally, we validat the method with a case study, and the knowledge base could be used in DSS. All these steps are shown in Figure 1.

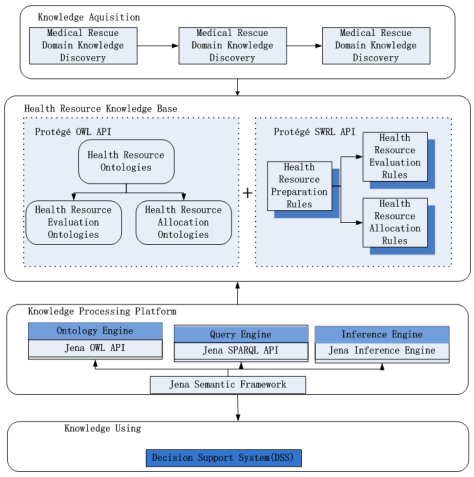


Figure 1. RESEARCH STEPS USED IN THE DEVELOPEMENT OF THE SHRKB

III. DEVELOPMENT OF ONTOLOGY-BASED KNWLEDGE BASE

A. Knowledge Acquisition

The knowledge acquisition phase includes three processes: knowledge identification, knowledge extraction, and knowledge authentication. Three types of knowledge sources are identified in the knowledge identification process:

- Published medical literature on healthcare resource preparation plans for crisis rescue (HRPPCR) or medical rescue guidelines, standard nomenclatures and medical rescue information models;
- Healthcare resource data from the history medical rescue tasks;
- Experiences of medical rescue practitioners and domain experts.1

The define of certain HRPPCR, is to design the type and quantity of the basic and specialized resources based on the possible distribution of the injuries' condition, category and portion injury[16,17]. The basic health resources(BHR) are for general injuries which always occur in most crisis rescuing. For example, the health resources for hemostasis, bandaging and fixing. The special health resources(SHR) are for special injuries which occur in certain crisis rescuing. For examples, the medicine and equipments for burned injuries in fire disaster; the medicine used for epidemic disease after flood disaster; etc.

In the processes, we gain a general view of the plan define workflow. We use the UML activity diagram to describe the workflow, as shown in Figure 2(taking BHR preparation as an example).

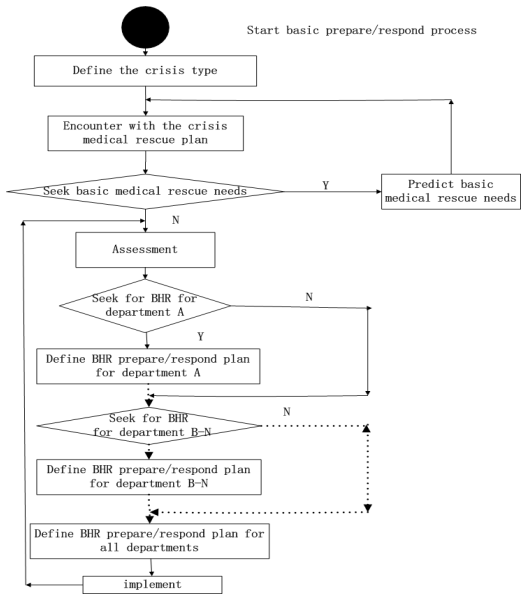


Figure 2. BHR PREPERATION WORKFLOW

Project in the National Science & Technology Pillar Program during the Twelfth Five-year Plan Period (No. 2015BAK10B01); Army logistics key projects(#BHJ12L007, # CHJ11J016).

The define of basic HRPPCR is a step-by-step process. Each department of the medical rescue team has a certain function. For example, the triage department mainly responds for first-aiding, sorting and allocating further aids on the basis of medical treatment needs. For another example, the operating department is mainly responded for performing surgery on the injury if necessary. These two departments are of core importance in a medical rescue team and need different types and quantity of health resources based on both the function of the departments and the workloads (medical rescue needs) predicted in the first step. So, we define the basic HRPPCR from department A to department N based on the rescue task needs. The whole basic HRPPCR is formed at the end of the process.

B. Information semantic extraction

We extract information semantics from these class diagrams using the meta-ontology approach[18]: concepts are expressed as classes; attributes and relationships are expressed as properties; decision algorithms are expressed as rules, as shown in Figure 3.

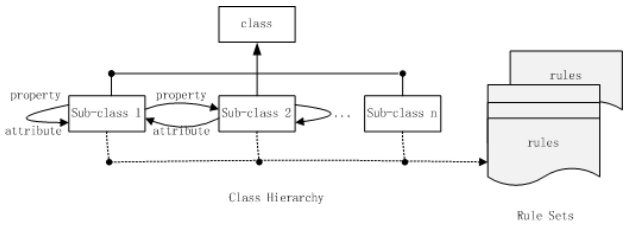


Figure 3. SEMANTIC EXTRACTION USING THE META-ONTOLOGY APPROACH

Close consultation with domain experts are involved in the concurrent knowledge authentication process to examine the necessity of extraction and the correctness of extracted information semantics.

C. Knowledge Representation

Knowledge representation focuses on the construction of the semantic health resource knowledge base (SHRKB), including the ontology of medical rescue domain knowledge and an expression repository of rules and queries. We develop SHRKB on Protégé, an open-source ontology editor and framework from Stanford University. In this paper, we use the development of the knowledge base of health resource evaluation as an example.

1) Semantic health resource knowledge ontology(SHRKO)

SHRKO is the backbone ontology of SHRKB to represent health resource domain knowledge. We develop SHRKO following the ontology developing guide on, main steps include: defining classes, defining the properties of classes, defining the linkages and creating instances.

Following a top-down design pattern, we define four top-level classes in SHRKO, i.e., Data, Model, Expressions and Rules. Below is a brief description of each class:

*a) **1.** Data.* **Data** models the standard data types defined in HL7 Version 3 Data Types Abstract Specification(HL7 v3.0 Data Types Specification.).

*b) **2.** Model.* ***Model*** is compliant to the model definition of the HL7 RIM Version 2.41 (HL7 Reference Information Model.), involving three main classes, ***Act, Role,*** and ***Entity,*** linked together with three association classes: ***ActRelationship, Participation*** and ***RoleLink***. ***Act*** represents all the acts, performed and recorded by responsible actors in a healthcare workflow; ***ActRelationship*** is a directed association between a source act and a target act; ***Entity*** represents physical thing, group of physical things or an organization capable of participating in an act while in a role; ***Role*** is a competency of an that plays the role as identified, defined, guaranteed, or acknowledged by the entity that scopes the role; ***Participation*** is an association between an act and a role; ***RoleLink*** is a connection between two roles expressing a dependency between those roles.

*c) **3.** Expression.* ***Expression*** is the linkage class between SHRKO and SHRER. We defined the following properties to represent linkage semantics: ***hasExprId*** is the expression identifier in SHRKO, ***hasExprFile*** is the pointer to the external expression file, ***hasTriggerSource*** marks the trigger source of the expression, either time-based, event-based or user defined, ***hasExpression*** is a textual description of the expression body.

*d) **4.** Rule.* ***Rule*** is the standard to describe the evaluation or distribution of the health resource. We defined the following properties to direct the making of rules in the rule base: ***Evaluation*** is the rule to evaluate the resource needs based on the task workloads or the prediction of the wounded types and quantity, ***Distribution*** is the rule to determine the health resource's distribution between departments or different medical rescue teams.

Based on this, we take these top-level classes as superclass and constructed subclasses layer by layer. Thus, we describe the conceptions of the sub-node of the knowledge system. Meanwhile, we add relevant attributes and linkages to the subclass. We also reuse existing domain ontologies if necessary, e.g., Systematized Nomenclature of Medicine-Clinical Terms(SNOMED-CT), Logical Observation Identifiers Names and Codes(LOINC), International Classification of Diseases-10(ICD-10). These are international standard medical term system. Using these ontologies to extend the semantic of our meta ontology could make the ontologies we developed interconnected with the international standards.

we use the development of the knowledge base of health resource evaluation as an example to show the creating of instance.

The objective of constructing the health resource evaluation knowledge base is to provide standards for quick health resource preparation based on different situation. The final plan based on this could be more fit for the mission needing without too much waste. Thus, the medical rescue team could bring more useful health resource with them to save more lives

making the limited transport capacity to the maxim effectiveness. The main factors to be considered are the number of wounded persons or people in disease, the possible distribution of the different injuries' condition and category, the number of medical rescuer. Health resources largely include medicine(M) resources (mainly pharmaceuticals), medical consumable goods, medical devices and medical instruments. The medicine resources and medical consumable goods are non-reusable resources. Medical devices and medical instruments are reusable resources. The main attributes of the resource were category (Kind) and resource quantity (Number).

Based on the analysis above, we construct the meta ontology to describe the standards of health resource evaluation, including four superclass as ***Data, Model, Expression and Rule***. Based on the meta ontology, we add constraint condition and supplementary condition to reify the subclass. Main subclass include: Medicine Class, describing basic information of medicine, including the attributes of name, ID, Coding System, Frequency, Dosage Unit, Dosage Daily, Drug Interaction, etc.; Device Class, describing basic information of medical devices, including the attributes of name, ID, Coding System, Max workload, Length, Height, etc.; Consumable goods Class, describing basic information of medical consumable goods, including the attributes of name, ID, Coding System, Qualification, etc.; Instrument class, describing basic information of medical instrument, including the attributes of name, ID, Coding System, Function, Assort, etc.; Injury Condition Class, describing specific condition of wounded, including the attributes of injury portion of body, injury type, injury pattern and injury trend; Distribution class, describing specific distribution of the resource between different departments or rescue teams; Term class, including referenced international standard medical term system, e.g. SNOMED-CT, LOINC, ICD-10, etc. Finally, we use OWL to describe the meta ontology and add the ontology instances to form the standard ontology as showed in figure4.

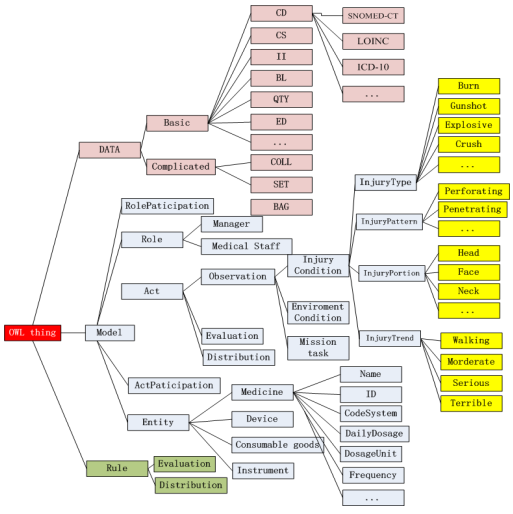


Figure 4. ONTOLOGY MODEL OF HEALTH RESOURCE EVALUATION

In Figure4, we use different color to mark different classes. The yellow marks are instances of certain subclass.

In addition, we import the SWRL temporal ontology to enhance the temporal expressivity of SHRKO with temporal models and built-ins.

2) Semantic health resource expression repository (SHRER)

SHRER is designed to extend SHRKO expressivity with expressions so that it can be queried and reasoned over for knowledge-based decision making. We consider two kinds of expressions: queries and rules. Queries are written in simple protocol and RDF query language (SPARQL). Rules are encoded in Jena rule format.

D. Knowledge Validation

We develop a prototype semantic-based health resource preparation decision support system (S-HRPDSS) in Java to enable SHRKB-based preparation decision making. The proposed method is validated in a case study of the overseas humanitarian medical assistance mission executed after a disaster by the hospital ship Peace Ark, a member of China's People's Liberation Army (PLA) Navy.

We instantiate the SHRKB with the domain knowledge from the Chinese marine medical guideline(CMMG)[19] and the standards of medical first aid in marine rescue mission from Advanced first aid afloat (Fifth edition)[20]; history marine war medical rescue data contained in the CMMG summaries, electrical medical records from "Mission Harmonious 2011", "Mission Harmonious 2013"and "Mission Harmonious 2014", as well as electrical medical records from emergent department of Navy General Hospital since 2010 are then imported into SHRKO as input fact data; given health resource related data(i.e. injury's or disease's condition, category) and rules from SHRER, we can use the Jena inference engine to derive new facts about the health resource preparation(evaluation or preparation plan recommendations).

Then we use cluster analysis, a statistics method to make plan on certain resource types to cope with the mission needs, based on normal knowledge base and semantic knowledge base to find the differences.

We define the health resource (HR) was defined as: HR=（M, C, D and I）M stands for medicine, C stands for medical consumable goods, D stands for medical Device, I stands for medical instruments. K stands for Kind, the main index in a health resource preparation plan. The cluster analysis is used to determine K as follows.

$HR_{ij} = (N_{ij} K_{ij}) whereby\ i = \alpha \beta \gamma\ j = M C D I$ and $K_{\alpha}, K_{\beta} \subset K$

K_{ij} is to describe all categories of resources; Nij is the index to describe the quantity of each kind of resources in the K_{ij}. α is for basic health resources, β is for special health resources. M stands for the medicine resources (mainly pharmaceuticals) , C stands for medical consumables, D stands for medical devices,

I stands for medical instruments.

The determination of K: K is associated with either the type of crisis or the function of different department in the medical rescue team. So the correlation coefficient R can be used for resource category clustering. Xi stands for the frequency use of a certain kind of the resource in "n" samples. When K_α is determined, the samples will be the special health resources used in the rescue task. When K_β is determined, the samples will be the health resources used in a certain department of the rescue team. X_j stands for the frequency of occurrence of a certain medical rescue tasks in "n" samples. Stronger correlation coefficient reveals a closer link between the resource category and the medical treatment function.

$$R_{ij} = \frac{\sum_{k=1}^{n} (X_{ki} - \bar{X}_i)(X_{kj} - \bar{X}_j)}{\sqrt{\left[\sum_{k=1}^{n} (X_{ki} - \bar{X}_i) \sum_{k=1}^{n} (X_{ki} - \bar{X}_i)^2\right]}}$$

IV. RESULTS

A total of 270 classes (4 top-level classes and 266 subclasses), 387 properties (319 object properties and 68 data type properties) and 219 rules are created in SHRKB.

The validation results from the case study of the overseas humanitarian medical assistance mission executed after a disaster by the hospital ship is demonstrated in Table 1. N stands for the health resource preparation plan clustered by the cluster analysis based on normal medical knowledge base (normal plan for short). S stands for the health resource preparation plan clustered by the cluster analysis based on our semantic health resource knowledge base(SHRKB) (SHRKB plan for short). A stands for the mission actual consumption of the health resource. (K stands for the set of health resource categories, |K| stands for the number of the medical resource categories.)

Table 1- Comparison of the resource categories of normal, actual consumption and SHRKB results.

	$\lvert K(N)\rvert$	$\lvert K(A)\rvert$	$\lvert K(S)\rvert$	$\lvert K(A) \cap K(N)\rvert$	$\lvert K(A) \cap K(S)\rvert$
Consumables	80	56	48	24	8
Medicine	243	167	140	76	27
Equipment	22	20	19	2	1
Instrument	52	52	52	0	0

As it is presented, the resource categories in the normal plan 100% met the requirement of the actual consumption. However there were more than 30% of the non-reusable resource categories in it remained unused during the mission. On the other hand, the inventory of 6 kinds of the non-reusable resources in the normal plan, especially some key resources, would be used up during the mission which indicating that the categories of resources are not adequate for the actual consumption. In the normal plan it appears that the surplus and deficiency could exist side by side.

The categories of the reusable resources in the SHRKB plan meet the requirement of the actual consumption at a more than 95% level. The categories of non-reusable resources meet the requirement at a level exceeding 80% mark. The categories which didn't appear in the plan but is present in the actual consumption are concentrated in some anti-infection medicines or anodyne which are used in a narrow range. These resources could be replaced by others resources which have a wider range of application. The SHRKB plan could meet the actual consumption with less categories of resources. Thus, this plan would be more simple and effective to health resource preparation in emergent case.

From this rough comparison , we could see the results made by the same decision method running on the SHRKB are better. It demonstrates the benefits of the combination use of the HL7 RIM and ontology.

We are still developing a DSS for health resource preparation in marine emergent case. The system prototype is showed in figure 5. This system could help managers to make a quick decision on health resource preparation both in category selecting, quantity calculating, supplement planning and better routes comparing. Thus, the medical rescue team would be ready to set off in a short time and cure injuries in the best time. The right part of the system shows the basic inputs including ship scheme, mission information, departments properties, injuries or disease sets. The up-left part shows the health resource allocation plan in the history case which matches the input condition on the left. The down-left part shows the health resource allocation plan recommended by the system through a series of OM calculating.

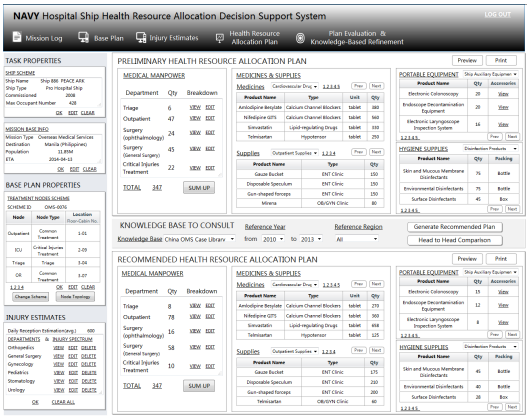


Figure 5 PROTOTYPE OF DSS FOR HEALTH RESOURCE PREPARATION IN EMERGENT CASE.

V. DISCUSSION AND CONCLUSION

In this paper, we present a new approach to enhance effective use of DSS in health resource preparation/respond by developing SHRKB, a health resource knowledge base based on rules of HL7 RIM and ontology. HL7 RIM offers a static model of health care information, which includes classes and state-machine diagrams and is accompanied by use case models, interaction models, data type models, terminology models, and other types of models based on which domain-specific information models are then derived through a series of constraining or refinement processes. Ontology provides a

standard model for data and knowledge representation.

The result of the case study using the same method running on different knowledge base shows, SHRKB could support the method come to more effective plan than normal knowledge base does. This demonstrates our research's advantages as follows: (1) the unified representation formalism for both mission data and domain knowledge to facilitate the semantic interoperability between DSS and healthcare information systems; (2) semantic-based integration and reuse of computer technologies and domain knowledge for convenient knowledge base maintenance; (3) knowledge-based inference for dynamic health resource preparation for crisis relief.

Still, our research has some imperfections. For example, if we want to develop DSS based on SHRKB, the validation of SHRKB is not enough. Considering the complication of the health resource preparation for the crisis, the data and knowledge acquisition are not enough. The evaluating of the health resource preparation plan is not enough. The data and knowledge acquisition is not enough because crisis rescuing is always in a hurry and there's hardly time for the rescuers to collect data or knowledge on purpose.

So, our future work will focus on: (1) develop DSS for health resource preparation based on SHRKB; (2) investigation of knowledge based preparation plan evaluation methods; and (3) integration with machine learning techniques for automatic knowledge acquisition and close-loop knowledge management.

ACKNOWLEDGMENT

This study is supported by the Project in the National Science & Technology Pillar Program during the Twelfth Five-year Plan Period (Construction Programming of emergency management mechanism and emergency plan system of towns, Grant No. 2015BAK10B01); Army logistics key projects(#BHJ12L007, # CHJ11J016).

REFERENCES

[1] Maughon JS An inquiry into the nature of wounds resulting in killed inaction in Vietnam. Mil Med , 1970, 1 : 13

[2] Sikic N, Korac Z, Krajacic I ,et al War abdominal trauma: usefulness of penetrating abdominal trauma index, injury severity score, and number of injured abdominal organs as predictive factors. Mil Med, 2001, 166: 226-230.

[3] Toregas, C., Swain, R.& ReVelle, C.et al. The Location of Emergency ServiceFacilities. Operations Research, 1971,19(6), 1363-1373.

[4] Badri, M. A.,Mortagy, A. K. & Alsayed,C. A. A multi-objective model for locating fire stations. European Journal of Operational Research,1998(110),243-260.

[5] Balcik,B. & Beamon,B. M. Facility location in humanitarian relief. International Journal of Logistics: Research and Applications, 2008, 11(2), 101-121.

[6] Rawls,C. G. & Tumquist, M. A. Pre-positioning of emergency supplies for disaster response. Transportation Research Part B,2010(44),521-534

[7] Ozbay, K.,Xiao,W. & lyigun, C.et al. Probabilistic Programming Models for Response Vehicle Dispatching and resource allocation in t raffic incident. management.I&SE: Rutgers University, 2004, 4-14.

[8] Karaman, M. M., Elmaghraby, W. & Salman,F. S. Relief Aid Stocking Decisions under Cooperation of Emergency Response Agencies,2009,1-4.

[9] Shetty, R. S. An event driven single game solution for resource allocation in a multi-crisis environment. Doctoral Dissertation, University of South Florida,2004.

[10] Gupta, U. Multi-event crisis management using non-cooperative repeated games,Doctoral Dissertation, University of South Florida, 2004.

[11] Sheu, J.B.,Chen, Y. & Lan, L. W. A novel model for quick response to disaster relief distribution. Proceedings of the Eastern Asia Society for Transportation Studies, 2005 (5), 2454-2462.

[12] Studer R, Benjamins V.R., Fensel D, Knowledge engineering: principles and methods. Data & Knowledge Engineering, 25(1), pp. 161-197, 1998.

[13] Collier N, Kawazoe A, Jin L, et al. A Multilingual Ontology for Infectious Disease Outbreak Surveillance: Rationale, Design and Challenges[J]. Journal of Language Resources and Evaluation, Springer, 2007,(40):3-4.

[14] Unified Medical Language System Semantic Network Documentation[EB/OL]. [2016-12-20].http://www.nlm.nih.gov/research/umls/meta3.html

[15] Schadow G, Russler D.C., Mead C.N., et al,Integrating medical information and knowledge in the HL7 RIM. Proc. of the AMIA Symposium, American Medical Informatics Association, pp. 764,2000.

[16] Kaiyu C, Yifeng C, Fen Z. Modularized support of medical supplies in military operations other than war. Academic Journal of Second Military Medical University 2012, 12:1351-1355.(Chinese)

[17] Hui J, Shengxin C, Guoquan R. Demand analysis and application of modularization theory in medical supplies for war preparedness. Academic Journal of Second Military Medical University 2005, 26:815-819.(Chinese)

[18] Z Hu, J.S. Li, T.S. Zhou, H.Y. Yu, Suzuki M, Araki K, Ontology-based clinical pathways with semantic rules. Journal of medical systems,36(4), pp.2203-2212, 2012

[19] Jiyao Yu, Zhengguo Wang. Naval Surgury. People's Military Medical Press, 2014.(Chinese)

[20] Peter F. Eastman, John M. Levinson. Advanced first aid afloat (Fifth edition)[M].Cornell maritime press, Maryland,2004.

.

Constant Evaluation of L2 Students' English Writing Ability

Li Li
Shanghai University of Political Science and Law
Shanghai, China
e-mail: lily2211@126.com

***Abstract*—Writing teaching is an indispensible part of college English teaching in China. Compared with L1 students, the writing teaching of L2 students is much more challenging. Recent years, Automatic Evaluation System (AES) has been more frequently employed to score students' essays by teachers. However, AES cannot replace teachers for the following shortcomings. 1) The essay evaluation is not very precise; 2) It cannot truly reflect a student's real writing ability based on independent scores. To solve the problems, this paper proposes a method of constantly evaluating L2 student's real writing ability, which evaluates, compares and analyzes several essays written by a student within a certain period. First, the framework of constant evaluation of writing ability is proposed, which consists of a single essay evaluation and a timeline-based evaluation. Next, several aspects of automatic evaluation of a single essay are improved. Then, a timeline-based writing ability evaluation method is proposed based on the knowledge graph. Finally, the experiments are conducted, the results of which show that the proposed method is effective in evaluating a student's real writing ability.**

Keywords-writing evalution; AES; auto scoring; L2 students; timeline-based evalution; knowledge graph

I. INTRODUCTION

Writing teaching is an indispensible part of college English teaching in China. Compared with L1 students, the writing teaching of L2 students is much more challenging. It is apparent that L2 students' English proficiency is lower than L1 students'. Each basic element of an article may be a barrier to writing for L2 students. That is to say, there will be problems of the choice of words, sentences, paragraphs or discourses. Therefore, it has been widely acknowledged that English writing teaching is a rather tough and challenging task. In order to increase the efficiency of evaluating students' essays and decrease the workload of teachers, Automatic Evaluation System (AES) has been more frequently employed in the writing teaching to help teachers to evaluate students' essays and turned out to be effective[1].

However, AES can't provide a precise evaluation because of the limitation of text understanding technologies, so it is impossible for AES to replace teachers to evaluate students' essays. What's more, the score of a single essay can't truly reflect a student's real writing ability. In the long run, if students depend much on it, AES may have a negative impact on their learning plan. In effect, a student's writing ability is influenced by diverse factors, apart from their own knowledge. For example, in different conditions, a student will present obviously different writing abilities. In a good condition, the choice of words, sentence-making and sentence variety will be at the high level; otherwise, they will be poor, which shows the volatility of one aspect of his/her abilities in writing. It shouldn't become a focus of teachers if a student shows that his/her abilities decrease temporarily in one aspect. On the contrary, it will have a negative effect on the student's learning initiative if teachers highlight this problem. Especially in China, there are a large number of students, it is rather hard for teachers to follow every student and constantly evaluate students' writing ability.

To solve the problems, this paper proposes a method of constantly evaluating L2 student's real writing ability, which evaluates, compares and analyzes several essays written by a student within a period. First, the framework of constant evaluation of writing ability is proposed, which consists of a single essay evaluation and a timeline-based evaluation. Next, several aspects of automatic evaluation of a single essay are improved. Then, a timeline-based writing ability evaluation method based on knowledge graph is proposed. Knowledge graph of writing ability is constructed to make a comprehensive description of a student's writing ability, including use of words, choice of sentences, coherence, logic etc. By analyzing the changes of knowledge graphs with time going on, the real writing ability of a student can be tested and obtained. Finally, the experiments are conducted and the experimental results show that the proposed method is effective in evaluating a student's real writing ability.

The rest of this paper is organized as follows: section II introduces related work, and section III puts forward a framework of constantly evaluating one's writing ability. Section IV and section V discusses the method of evaluating a single essay and the method of constantly evaluating one's writing ability respectively. In section VI, the experiment is performed to verify the proposed method and the last section draws a conclusion.

II. RELATED WORK

AES is defined as the computer technology that evaluates and scores the written prose [2]. With the growing development of computer technology, AES systems have been improved a lot and are being improved. In order to make the large-scale essay scoring process more practical and effective, Project Essay Grader (PEG) was developed by Ellis Page upon the request of College Board [3]. It utilizes proxy measures to evaluate the quality of essays. But it has been criticized for ignoring the semantic aspect of essays and focusing more on

DOI reference number: 10.18293/SEKE2017-149

the surface structures [4][5]. With the advance of computer technology, more AES systems, such as Intelligent Essay Assessor (IEA), the Electronic Essay Rater (E-Rater) were developed to meet the requirements. IEA analyzes and scores an essay using a semantic text analysis method called Latent Semantic Analysis (LSA). It is claimed that unlike other AES systems, IEA's main focus is more on the content-related features rather than the form-related ones. However, this doesn't mean that IEA offers no feedback on formal aspects, i.e., grammar and punctuation, in an essay. However, the system doesn't evaluate the creativity and reflective thinking. E-Rater was developed by the Educational Testing Service (ETS) to evaluate the quality of an essay by identifying linguistic features in the text [6]. E-Rater uses natural language processing (NLP) techniques, which identify specific lexical and syntactic cues in a text, to analyze essays [5]. Later, artificial Intelligence (AI) was introduced to the development of AES systems. IntelliMetric, developed by Vantage learning, is known as the first essay-scoring tool that was based on AI. Like, E-Rater, IntelliMetric relies on NLP, which determines "the meaning of a text by parsing the text in known ways according to known rules conforming to the rules of English language" [7]. Another AES system, named My Access, is known as the instructional application of IntelliMetric. My Access is a web-based writing assessment tool that relies on Vantage Learning's IntelliMetric automated essay scoring system. The main purpose of the program is to offer students a writing environment that provides immediate scoring and diagnostic feedback, which allows them to revise their essays accordingly and motivates them to continue writing on the topic to improve their writing ability. ETS' Criterion, a web-based instructional writing tool, uses the E-Rater engine to provide both scores and targeted feedback. It allows students to improve their writing skills while working independently with immediate, detailed feedback on grammar, spelling, mechanics, usage, and organization and development.

III. FRAMEWORK OF CONSTANTLY EVALUATING STUDENTS' WRITING ABILITY

According to the above analysis, it is a gradual process to enhance students' writing ability and there exist ups and downs in the writing quality with individual factors and changes of the external surroundings outside. L2 students will show different features of changes in their abilities of different writing elements with the time passing. For example, there remains greater influence on diction (choice and use of words). As to different themes and styles, there are great differences. If they haven't written an essay of the same subject for a long time, students' ability of using the words of the subject will obviously decrease. Comparatively speaking, the ability of writing arrangement is rather stable. As long as they keep practicing writing, students can maintain this ability and are likely to enhance it. So considering such factors as fluctuation of writing ability, the evaluation of a single essay can't reflect a student's real writing ability objectively and comprehensively.

The paper proposes the method of constantly observing and evaluating a student's writing ability, whose framework is shown in Fig. 1 and mainly consists of two modules.

1) Single essay evaluation module

This module realizes the evaluation of a single essay, which is similar to the function of AES. The difference between them is that this module also provides additional data for the writing ability evaluation module so as to build the forgetting curve of the student.

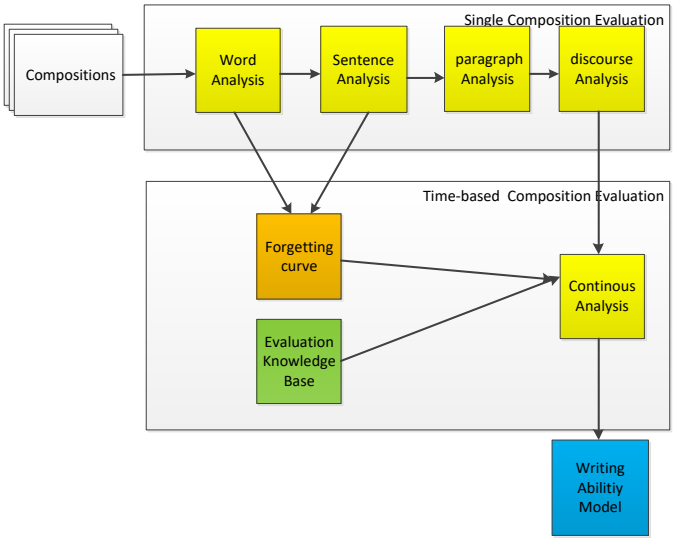


Figure 1. Framework of constant evaluation of writing ability

This module is composed of word evaluation, sentence evaluation, paragraph evaluation, discourse evaluation and an overall evaluation by summarizing these evaluations of writing elements. The overall score of an essay gives a direct and straightforward evaluation to students, while the evaluation of writing elements can present students their weak points in certain aspect, which has a much more instructive effect on writing teaching.

2) Writing ability evaluation module

This module is to evaluate students' writing ability comprehensively within a certain period. Namely, based on evaluations of several essays within a certain period, the timing analysis is adopted to analyze different writing ability evaluation indexes and then the knowledge graph of writing ability is built.

The indexes of writing ability are related to each other, so it is not likely to improve the overall level of the students effectively only by intensifying the training of one element. For instance, a student leaves much room to improve his/her ability of making a sentence. If he/she only practices making a sentence, he/she can't really improve his/her overall writing ability, for how to make a sentence has much to do with such elements as words, phrases, grammar, sentence patterns.

Knowledge graph is a model used to show the relation between knowledge points, which can offer both the overall view and the detailed view. Therefore, the writing ability is modeled by knowledge graph, which can also show the writing ability of a student from both the overall and the detailed aspects.

IV. SINGLE ESSAY EVALUATION

Compared with teachers' evaluation, AES has the following shortcomings when just used to evaluate a single essay: 1) the precisions of some evaluation indexes are low; 2) some evaluation indexes cannot be assessed by AES. However, the distinguishing advantage is its high speed, which is suitable for the heavy task of constantly evaluating a large number of essays of the students by using our method.

A. Evaluation indexes

Taking both the difficulties in the implementation and the precision of AES into consideration, we select some feasible and practicable evaluation indexes to evaluate a single essay evaluation, which are shown in Table I.

TABLE I. ESSAY EVALUATION INDEXS

One class index	Two class index	Weight
1.Word & phrase	1.1 Spelling	0.3
	1.2 Grammar	0.3
	1.3 Vocabulary	0.5
2.Sentence	2.1 Punctuation	0.2
	2.2 Sentence structure	0.3
	2.3 Sentence grammar	0.3
	2.4 Sentence pattern	0.5
3.Pragraph	3.1 Sentence coherence	0.6
	3.2 Topic relevance	0.8
4. Discourse	4.1 Ideas	0.7
	4.2 Organisation	0.6
	4.3 Pragraph coherence	0.6
	4.4 Theme relevancy	0.8

B. Evaluation method

Among the above-mentioned evaluation indexes, some can be easily realized by means of computer algorithm. For example, the evaluation of spelling and grammar can reach high precision with the support of the dictionary and grammar corpus. But evaluations of coherence, unity and transition are not easy to carry out. Some evaluation indexes in our method are discussed as follows.

1) Vocabulary: To vocabulary, we focus on the evaluation of the breadth and depth of the use of words. Taking into account the actual situation of Chinese students, we select ten levels of vacabularies as the standards, shown in Table II, whose difficulties increase gradually.

The ability of choosing and using the words is calculated based on the distribution of words in the essay, which is calculated as

$$v = \sum_{i=1}^{n} (\frac{l_i}{m} \times w_i) = \frac{1}{m} \sum_{i=1}^{n} (l_i \times w_i), \quad (1)$$

where, m is the number of words in the essay, l_i is the number of words in the level i vocabulary, and $\frac{l_i}{m}$ is the ratio of level i in the essay. The ability of using the words, v, is the weighted sum of the use of words at all levels. It is valuable to know the changes of a student's v within the given time for the purpose of evaluating his ability of using the words.

TABLE II. DIFFICULTY LEVELS OF VOCABULARY

Level #	Vocabulary	Weight
1	College English Test Band 1	0.1
2	College English Test Band 2	0.2
3	College English Test Band 3	0.3
4	College English Test Band 4	0.4
5	College English Test Band 5	0.5
6	College English Test Band 6	0.6
7	Test for English Majors Band 4	0.7
8	Test for English Majors Band 8	0.8
9	TOFEL	0.9
10	GRE	1.0

2) Sentence Pattern

The variety of sentence patterns is viewed as a main evaluation index. If different sentence patterns are employed to express one's opinions in an essay, the whole will be richer. A succession of simple sentences may be jerky and choppy, a succession of loose sentences relaxed or even slovenly and a succession of periodic sentences formal, stiff and difficult to follow. Too many sentences of the same pattern following one another are at least monotonous.

According to different standards, sentence can be divided into different types. According to their use, sentences are declarative, interrogative, imperative or exclamatory. According to their structure, sentences are simple, compound, complex or compound-complex. From the rhetorical point of view, sentences are loose, periodic and balanced.

In the English writing, the method of evaluating the variety of sentence patterns is used to calculate the distribution of different sentence patterns in an essay. The calculation method is

$$s = \frac{sp}{sn} - \frac{1}{sm} \sum_{i=1}^{m} \mathrm{N}sp_i \ , \quad (2)$$

where, $\frac{sp}{sn}$ denotes the ratio of sentence patterns used in the writing, sn and sp denotes the number of the variety of sentence patterns and the number of the variety of sentence patterns in the writing respectively; $\frac{1}{sm} \sum_{i=1}^{m} \mathrm{N}spi$ measures

repetition of sentence patterns, *sm* is the number of specific sentence patterns, nsp_i is the frequency of repeating the sentence pattern I, *sm* is the number of sentences in the writing.

3) Coherence

The coherence of an essay includes the text coherence and the semantic coherence.

Text coherence is a literary technique that refers to the meaningful connections that readers perceive in a written text. In other words, it is a well-written piece that is not only consistent and logical, but also unified and meaningful. It makes sense when read as a whole. The structure of a coherent paragraph could be general to particular and particular to general or any other format. In order to achieve the effect of coherence, proper transitions have to be employed which are used to make a connection clear [9]. The local coherence and global coherence of an essay can be evaluated by analyzing the transitional words and phrases.

Semantic coherence refers to the coherence of content, namely, the association of the whole passage from the beginning to the end. Given that L2 students don't usually write a long essay and each paragraph consists of several sentences, it is not easy to calculate the semantic coherence of sentences within a paragraph. This paper mainly focuses on the semantic coherence among paragraphs, the algorithm of which is described in Algorithm 1.

Algorithm 1: Semantic coherence evaluation of an essay
Input: C // a student's essay
Output: *sc* // semantic coherence

1) For each p_i in C do // p_i is the i[th] paragraph of C
2) extract keywords of p_i using TF-IDF
3) denote $p_i = \{k_1, k_2, ..., k_n\}$
4) End for
5) Mine association rules based on paragraphs and get the association rules set *ARs*
6) For i=1 to n do
7) Calculate the association degree sc_i between p_i and p_{i+1}
8) End for
9) $sc = \frac{1}{n-1}\sum_{i=1}^{n-1} sc_i$

In the algorithm, each paragraph is denoted by VSM (from step 1 to 4). And then, each paragraph can be seen as a transaction and the essay can be seen as a transaction set. Each keyword of a paragraph can be seen as an item. As a result, the association rule mining algorithm can work on it. In this step we can get the association rule set *ARs* in this paragraph. Step 7 finds out all the association rules in ARs which can bridge the neighboring paragraphs. Weighs of all the selected association rules can be summed to get the sc_i. In the end, the average of the semantic coherence between each pair of neighboring paragraphs is calculated and then viewed as the total semantic coherence of the essay.

4) Theme relevancy

What students write must center about a given theme, otherwise they will have to face the danger of straying away from the point. By calculating the similarity between each paragraph and the theme, we can obtain a value of theme relevancy. The method similar to Algorithm 1 is employed to show each paragraph's VSM, namely, $p_i = \{k_1, k_2, ..., k_n\}$. Meanwhile, the requirements of the writing can be represented by VSM, $theme = \{t_1, t_2, ..., t_n\}$.

The similarity between p_i and *theme* can be calculated by COS method

$$sim(p, theme) = \frac{\sum_{i=1}^{n}(k_i \times t_i)}{\sqrt{\sum_{i=1}^{n}(k_i)^2} \times \sqrt{\sum_{i=1}^{n}(t_i)^2}} \quad (3)$$

And the total theme relevancy can be gotten by calculating the average of similarity of all paragraphs.

V. COMPREHENSIVE EVALUATION OF WRITING ABILITY BASED ON TIMING ANALYSIS

A. Knowledge Graph of writing abliltiy

To offer students their global evaluation and specific evaluation on their writing ability, the paper introduces knowledge graph to present students' writing ability, shown in Figure 2. In the figure, circle refers to evaluation indexes at different levels. Yellow circle(dotted box) indicates level 0, which is students' global evaluation; green one(dashed box) refers to level 1, which is similar to the first row of Table I. Gray one(solid box) refers to lever 2, similar to the second row of Table I. The size of the circle shows the weight of the nodes, here referring to the score a student gains on the index. In order to directly demonstrate the continuous changes of students' writing ability, the model employs numbers from 0 to 100 to show the weight of nodes.

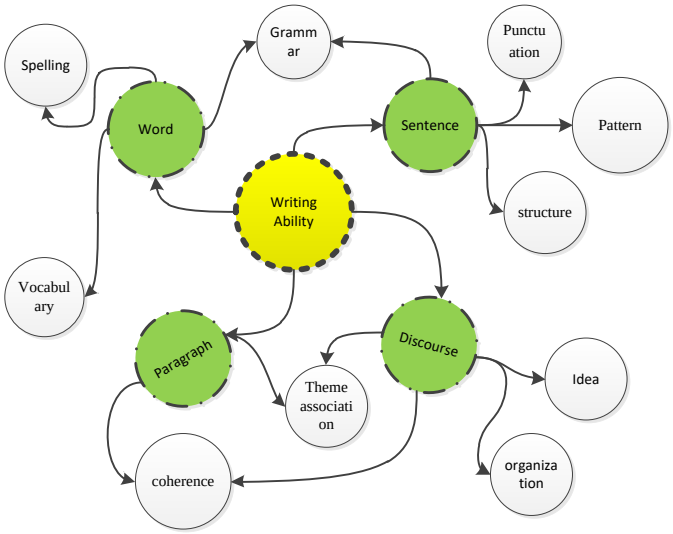


Figure 2. Knowledge graph of writing ability

In Figure 2 drawn from Table 1, the arrow shows the relevancy exists between the two evaluation indexes. As to

different students, their knowledge graph of writing ability is different, is listed as follows:

a) The size of nodes: The size of nodes indicates that students have different abilities in a certain aspect of writing. According to the size of nodes, students can do some targeted and specific training, which contributes to improving their overall writing ability.

*b) The length of sides:*The length of *edge* indicates that students have different connections between the indexes of writing ability. If he/she gets a low score on *an* index, he/she has to intensify his training of this index and relevant indexes, and then they will bear fruit.

B. Build the Knowledge Graph of writing abliltiy

Because we are still keeping working on mining the relation between evaluation indexes, here we mainly discuss the method of calculating the node weight of knowledge graph of writing ability.

According to the above analysis, the evaluation of a single essay cannot be used to judge a student's writing ability. We must eliminate ill effects of some factors including the surroundings on students' writing ability from the perspective of time. Therefore, the model of a student's writing ability needs to save the historical evaluation data. The model of a student's writing ability is represented by several knowledge graphs with time label. Figure 3 presents the first level of evaluation index of a student's knowledge graph in the time sequence. This Figure shows the changes in his /her writing ability and provides historical data for us to calculate his/her next knowledge graph as well.

When analyzing a single index from the aspect of time, we will find the influence of such factors as the surroundings will be singular points in the writing ability curve, such as the fourth node shown in Figure 4. If the cycle is short, these singular points can be effectively eliminated by means of linear fitting. If the cycle is long, Detrended Fluctuation Analysis (DFA) will be adopted to eliminate the detrended fluctuation [8]. Detrended fluctuation analysis (DFA) is a method of determining the statistical self-affinity of a signal.

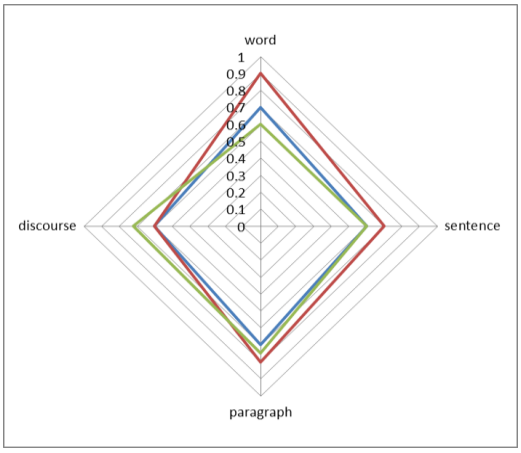


Figure 3. Time-based writing ability model

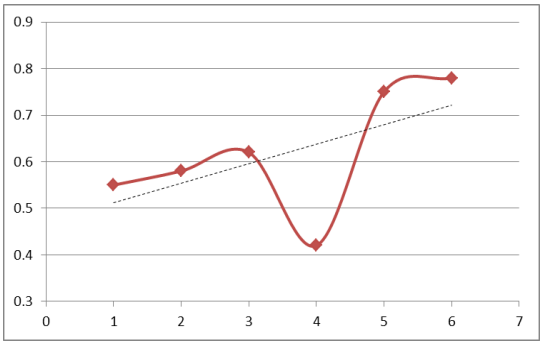


Figure 4. Time-based analysis of writing ability index

VI. EVALUATION

To verify the effect of the proposed method, the comparative experiment is conducted to compare the precision of evaluation of teachers with that of the proposed method.

A. Participants

Participants are involved in ten students and ten teachers. Twenty essays written by each in two semesters are regarded as experimental data. Ten teachers fall into two groups, five teachers in each. Among them, one is experimental group, the other is expert group. The experimental group mainly evaluates students' essays and their writing ability. While the expert group judges the evaluation results of teacher and the proposed method.

B. Procedures

1) Teachers in the experimental group adopt a method of global scoring to evaluate each essay in the sequence of time when students complete them. Meanwhile, they have to evaluate them based on the first level and second level indexes respectively.

2) Teachers in the experimental group require timing analysis of twenty essays of each student. Every four essays is seen as an observation point and builds a knowledge graph of writing ability, so there are five knowledge graphs totally.

3) The first two steps are repeated by computer based on the proposed method.

4) The teachers in expert group compare the evaluations of teachers in the experimental group with those given by computer.

C. Experimental Results and Analysis

The experimental results are shown in Figure 5 and Figure 6. Figure 5 shows the comparative experiment on the precision of writing evaluation. It must be pointed out that each essay should be evaluated by five teachers and finally take the average. The data in Figure 5 is the average of twenty essays of each student.

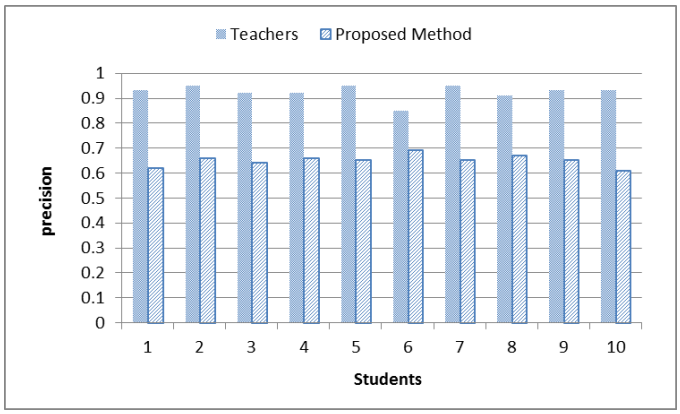


Figure 5. Comparison of scoring precision between teachers and the proposed method

Figure 6 shows the result of comparative experiments on the construction of knowledge graph. The comparative item is the precision of construction method. To make the results more general, the final knowledge graph of each student is constructed on the average of knowledge graphs constructed by five teachers independently.

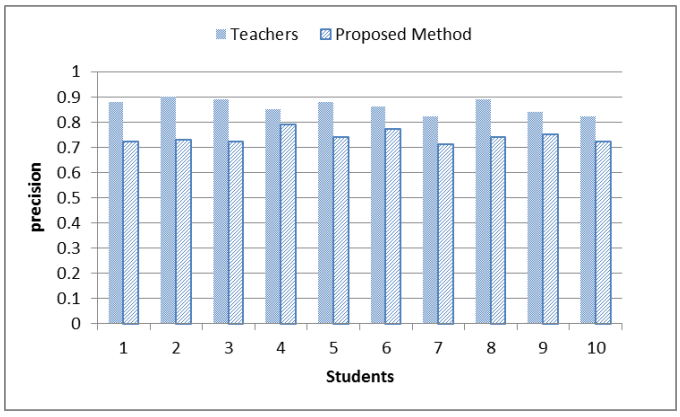


Figure 6. Comparison of knowledge graphs constructed by teachers and the proposed method

The results of comparative experiment show that the precision of teachers' evaluation is better than the proposed method, in terms of evaluation of a single essay or comprehensive evaluation of writing ability. But the results also indicate that the gap between machine scoring and manual scoring narrows when they conduct a comprehensive evaluation.

The experimental process still shows that the speed of teachers' evaluation is much slower than machines'. What's more, in the large-scale evaluation, the precision of manual evaluation will decrease. The results indicate that the precision of machine evaluation is bigger than 0.7, and the error is less than 0.15, compared to the accuracy of teachers. So in the large scale evaluation, machine can take the place of teachers.

VII. CONCLUSIONS

As to the problem of evaluation of L2 students' English writing ability, the method of constantly evaluating their writing ability is proposed and timing knowledge graph is constructed to reflect students' overall writing ability so as to have the knowledge of students' real writing ability. The experimental results show that the proposed method is effective in evaluating a student's real writing ability.

The future work is involved in carrying out a deep research on mining the relation of the indexes of knowledge graph. At the same time, the research on how to work out learning plan automatically by means of timing evaluation results is going to be carried out.

ACKNOWLEDGEMENTS

This research was financially supported by 2017 Scientific Research Project of Shanghai University of Political Science and Law.

REFERENCES

[1] Mcnamara, D. S., Crossley, S. A., Roscoe, R. D., Allen, L. K., & Dai, J. (2015). A hierarchical classification approach to automated essay scoring. Assessing Writing, 23, 35-59.

[2] M. D. Shermis, and J. C. Burstein, "Automated essay scoring: A cross-disciplinary perspective", Routledge, 2003.

[3] E. B. Page, "Project essay grade: PEG," Automated essay scoring: A cross-disciplinary perspective, pp.43-54, 2003.

[4] G. K.Chung, and H. F. O'Neil, Methodological approaches to online scoring of essays. Center for the Study of Evaluation, National Center for Research on Evaluation, Standards, and Student Testing, Graduate School of Education & Information Studies, University of California, Los Angeles,1997.

[5] K. Kukich, "Beyond automated essay scoring," IEEE intelligent systems, vol.15, no.5, pp.22-27,2000.

[6] J. C. Burstein, D. Marcu, S. Andreyev, and M. Chodorow, "Towards automatic classification of discourse elements in essays," In Proceedings of the 39th annual meeting of the Association for Computational Linguistics, pp.90-92, France, 2001.

[7] S. Elliot, "IntelliMetric: from here to validity. Automated essay scoring: A cross-disciplinary perspective," Automated essay scoring: A cross-disciplinary perspective, pp. 71-86, 2003.

[8] Peng, C.K.; et al. (1994). "Mosaic organization of DNA nucleotides". Phys. Rev. E. 49: 1685–1689. doi:10.1103/physreve.49.1685

[9] Duane H. Roen, "Coherence." Encyclopedia of Rhetoric and Composition: Communication From Ancient Times to the Information Age, ed. by Theresa Enos. Taylor & Francis, 1996

Knowledge Discovery Process for Description of Spatially Referenced Clusters

Giovanni Daián Rottoli [1,2,3], Hernán Merlino [3], Ramón García-Martinez [3,4 †]
[1] PhD Program on Computer Sciences. National University of La Plata. Argentina.
[2] PhD Scholarship Program to Reinforce R+D+I Areas. Technological National University. Argentina
[3] Information Systems Research Group. National University of Lanús. Argentina
[4] Scientific Research Commission - CIC Bs As, Argentina. Argentina
gd.rottoli@gmail.con, hmerlino@gmail.com

Abstract— **Spatial clustering is an important field of spatial data mining and knowledge discovery that serves to partition a spatial data set to obtain disjoint subsets with spatial elements that are similar to each other. Existing algorithms can be used to perform three types of cluster analyses, including clustering of spatial points, regionalization and point pattern analysis. However, all these existing methods do not provide a description of the discovered spatial clusters, which is useful for decision making in many different fields. This work proposes a knowledge discovery process for the description of spatially referenced clusters that uses decision tree learning algorithms. Two proofs of concept of the proposed process using different spatial clustering algorithm on real data are also provided.**

Keywords- Knowledge Discovery Process; Spatial Clustering; Regionalization; Decision Tree Learning; Spatial Data Mining.

I. INTRODUCTION

Clustering is an information mining and knowledge discovery task that can be used to partition a spatial data set to obtain disjoint subsets whose elements are similar to each other [1, 2].

There are three ways to consider spatial information in clustering analysis: (i) spatial clustering to find disjoint groups of spatial points using spatial attributes, non-spatial attributes or both; (ii) regionalization, adding a spatially contiguity constraint between spatial objects to spatial clustering, and (iii) point pattern analysis, to detect unusual concentrations of spatial points in some regions of the space [3].

This variety adds a level of complexity to the data mining task [3] and, because of that, new algorithms and methods for spatial clustering have been developed in recent years: REDCAP algorithms for regionalization; DBSCAN, NSCABDT, ASCDT, among others, for clustering of spatial points; and DBSCAN and RDBC for point patterns analyses, to name a few [3, 5-16].

However, these algorithms just make use of different strategies to generate groups or clusters between the different spatially referenced objects, but none of them serves to obtain, in a systematic way, the description of the automatically generated clusters in order to know which criteria were used to realize this activity.

For this reason, a knowledge discovery process, defined as a group of logically related tasks that form a set of information with a degree of value for the organization obtains knowledge pieces that generalize the previous information [1, 17, 18], is designed to generate decision rules on clustering results regardless of the selected approach to generate spatial clusters.

The remainder of this paper is organized as follows: Section 2 describes the problem derived from the analysis of the state-of-the-art. In Section 3, we propose a Knowledge Discovery Process to solve the problem described. In Section 4, two proofs of concept are presented using real data. Finally, conclusions derived from the research are outlined in Section 5.

II. PROBLEM DEFINITION

Many algorithms and methods were developed to discover spatially referenced clusters in any of its forms: spatial clusters, regions or point patterns. These algorithms make use of different heuristics and techniques to separate the spatial objects more similar to each other, according to a specific similarity function. As result of this task, for each spatial object, the cluster to which it belongs is specified. However, as mentioned before, these algorithms do not allow describing the automatically discovered clusters according to the attributes chosen for that activity.

This issue affects the decision makers that use this kind of data mining algorithms. When spatial clustering algorithms are used and because clusters are not relevant by themselves, the results obtained have to be analyzed and visualized to retrieve relevant information for decision making in a certain problem domain.

If there is no systematic way to conduct this activity, decision-makers should make an extra effort to interpret the results or should be necessary to spent more resources using experts to do this task previously.

For this reason, it is interesting to design a knowledge discovery process, such as those proposed in [1], which can be used to obtain partitions of the information mass in a systematic way and then describe each partition or cluster according to the values of the attributes of the data that belong to each of them.

DOI reference number: 10.18293/SEKE2017-013

III. PROPOSED SOLUTION

García-Martinez *et al.* (2013) proposed a knowledge discovery process for Group-Membership Rules to identify the conditions of membership to each of the classes of an unknown partition *a priori*, but existing in the available information bases of the problem domain. This process makes use of Top Down Induction of Decision Tree algorithms (TDIDT) on the result of applying a clustering algorithm, e.g. Self-Organizing Maps (SOM), on transactional data to determine the conditions to belong to a group. Based on this previous work, a knowledge discovery process for description of spatially referenced clusters is proposed in this paper following the same concept.

This process, as mentioned before, serves to find spatial clusters in any of its forms (i.e., regionalization, clustering of spatial points and point pattern analysis), and find the rules that describe the characteristics of each of them, based on the data attributes selected to be used for clustering. Figure 1 shows the proposed process using Business Process Model Notation [19].

The process, as can be seen in Figure 1, takes a set of spatially referenced data as input represented in different formats such as, *inter alia*, plain text, databases, tables and geographic information system maps. These data are integrated to a table comprised of the object identifier, spatial attributes (e.g. object location), and non-spatial attributes such as object properties according to the problem domain.

The integrated data are used for cluster discovery process [1]. For this purpose, it is necessary to select a type of spatial cluster among the aforementioned types: regionalization, clustering of spatial points or point patterns analysis, according to the problem domain, and choose appropriate algorithms in each case.

This paper proposes the use of any of the REDCAP algorithms for region generation, because of their benefits over other regionalization algorithms [6]. In this case, it is necessary to provide contiguity constraints between the spatial objects as algorithm input.

On the other hand, both in the case of clustering of spatially referenced objects and point pattern analyses, it is suggested to use density-based algorithms[11,14], such as DBSCAN-like algorithms [11, 15, 16], DENCLUE [20], ASCDT [9] o DBSC [10], because of the same reason mentioned above.

In each case, the input attributes depend on the selected algorithm. After the clustering algorithm execution, it is necessary to create a new table comprised of the integrated information with a new attribute in which each row registers the cluster to which the spatial object belongs.

In the last step, a Decision Tree Learning algorithm, such as C4.5 [21] or Random Forest [22-24], to name a few, is used to generate the rules that describe the characteristics of each cluster. For this purpose, it is necessary to identify the input attributes and the target attribute beforehand: input attributes will be the non-spatial data attributes, and target attribute will correspond to the attribute added in the last step, which has information about the automatically generated clusters. The parameters for the decision tree learning algorithms have to be selected depending on the particularities of the data.

As a result of the proposed knowledge discovery process, a set of rules that describes the automatically generated clusters is provided.

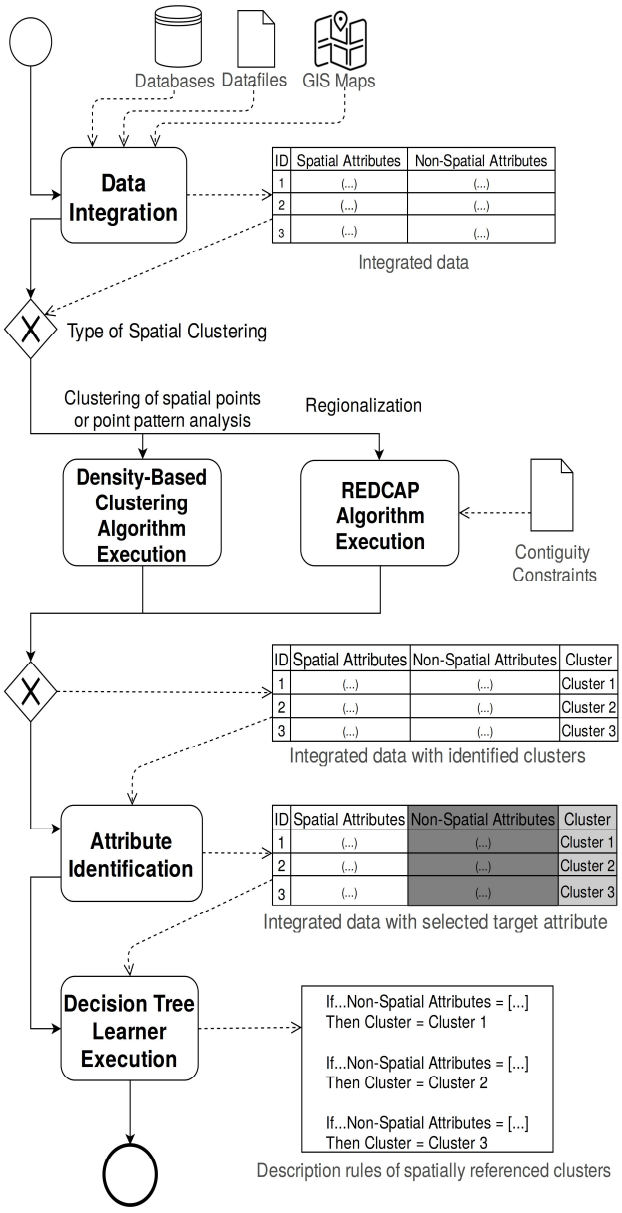


Figure 1. Knowledge discovery process for description of spatially referenced clusters

IV. PROOFS OF CONCEPT

This section includes two proofs of concept of the proposed process using real datasets obtained from different sources. In subsection 4.1 regionalization algorithms are used on demographic data, and in subsection 4.2 density-based clustering algorithms are applied to meteorological stations data.

A. *Demographic Data Regionalization*

In this proof of concept of the proposed knowledge discovery process, demographic data of 100 cities from the state of Iowa (IA) of the Unites States of America, updated on October 4th, 2016 and obtained from [26] were used for regionalization.

In the first step of the process, the selected data were integrated and normalized into a set of attributes, as shown in Table 1, being all of them numeric attributes.

Then, a regionalization algorithm from the REDCAP family of regionalization algorithms was selected: the First Order-SLK algorithms. This algorithm is not the REDCAP algorithm with a better behavior [6], but the simplest among them and, because of that, it was chosen to illustrate how the proposed process behaves. In this case, Delaunay triangulation was selected as contiguity criterion: two cities are contiguous if they are connected by an edge in the Delaunay triangulation of all the cities, using the Euclidean distance between them [25].

After the application of this algorithm selecting 6 as the number of desired regions to be obtained, the regions shown in Figure 2 using Voronoi diagrams [27] with each city as generator were discovered. Then, those regions were integrated in a single record with the original data.

Later, the Top Down Induction of Decision Tree algorithm C4.5 [20] was selected to generate the rules that describe the discovered regions, using Tanagra's implementation [28] and selecting the discovered region column in the integrated data as target attribute, and the demographic attributes "Population", "Households", "Median_income" and "Land_area" as input attributes.

As a result, we obtained 13 rules. For each region there is one or more rules than describe it. If the region is described with more than a single rule, it means that the region contains non-similar elements. Figure 3 shows the rules of the regions described with a single rule. On the other hand, the rules of the regions described with more than a single rule are shown in Figure 4.

As shown in Figure 3, regions 1, 2 and 3 are described with a single rule. Region 1 has more than 30 households, a population greater than 107 but lower than 2394, median incomes lower than USD72500 and a land area between 6236971.5 and 21839152 square meters. Region 2 has a very simple description: if the land area of the city is greater than 21839152 square meters, the city belongs to region 2. Finally, the rule that describes region 3 has a low confidence; it means that the cities that belong to it are not very similar, only 42.86% of the cities correspond to the description. The remaining 57.14% are not similar to each other and, because of that, there is no other rule that describes it.

The regions described with many rules, such as regions 0, 4 and 5 (Figure 4), can be divided into new regions to obtain more homogeneous sub-regions. For instance, region 5 is described with 2 rules with over 60% confidence: the first rule refers to cities with a land area between 4113504 and 6236971.5 square meters, a population between 107 and 2394, a median income lower than USD72500 and over 30 households, while the second one refers to cities with less than 30 households and a land area lower than 21 839 152 square meters. If more regions are specified, it is possible to divide the regions described using many rules into many different and homogeneous regions. However, this depends on the heterogeneity function used in the regionalization algorithm, which in turn depends on the domain of the problem.

TABLE I. DESCRIPTION OF THE ATTRIBUTES OF THE DEMOGRAPHIC DATA USED IN THE FIRST PROOF OF CONCEPT

Attribute	Description
Id	City Identifier
Latitude	City coordinates
Longitude	
Population	The total population living within city limits, using the latest US Census 2014 Population Estimates
Households	The total number of households within city limits using the latest 5 year estimates from the American Community Survey.
Median_income	The average (median) household income for the record using the latest 5 year estimates from the American Community Survey (USD)
Land_area	The area of land covered by the city in sq. meters.

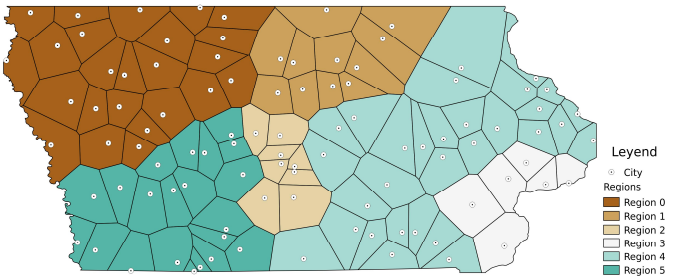


Figure 2. Regionalization of cities of the state of Iowa, USA

```
IF  "households" >= 30
AND "population" >= 107
AND "population" < 2394
AND "median_income" < 72 500
AND "land_area" < 21 839 152
AND "land_area" >= 6 236 971.5
THEN Region = 1  [Confidence = 66.67%]

IF "land_area" >= 21 839 152
THEN Region = 2  [Confidence = 83.33%]

IF "households" >= 30
AND "median_income" < 72 500
AND "land_area" < 1 011 569.5
AND "population" >= 245
THEN Region = 3  [Confidence = 42.86%]
```

Figure 3. Description rules of regions described with a single rule, obtained using the proposed process.

```
If "land_area" >= 1 011 569.5
And "land_area" < 1 427 927.0
And "median_income" < 72500.0
And "population" >= 106.5
And "households" >= 30
Then Region = 0 [Confidence = 85.71%]

If "population" < 245
And "population" >= 106.5
And "land_area" < 1 011 569.5
And "median_income" < 72 500
And "households" >= 30
Then Region = 0 [Confidence = 71.43%]

If "population" >= 522,5000
And "land_area" >= 2 579 852.5
And "land_area" < 4 113 504
And "median_income" < 72500,0000
And "households" >= 30,0000
Then Region = 0 [Confidence = 83.33%]

If "land_area" < 2 579 852.5
And "land_area" >= 1 760 551.5
And "median_income" < 72 500
And "population" >= 106.5
And "households" >= 30
Then Region = 0 [Confidence = 0,5556]

If "population" < 106
And "households" >= 30
And "land_area" < 21 839 152
Then Region = 4 [Confidence = 60%]

If "median_income" >= 72 500
And "population" >= 106.5
And "households" >= 30
And "land_area" < 21 839 152
Then Region = 4 [Confidence = 60%]

If "population" >= 2 394,5
And "land_area" >= 4 113 504
And "land_area" < 21 839 152
And "median_income" < 72 500
And "households" >= 30
Then Region = 4 [Confidence = 57.14%]

If "land_area" < 1 760 551.5
And "land_area" >= 1 427 927
And "median_income" < 72 500
And "population" >= 106.5
And "households" >= 30
Then Region = 4 [Confidence = 77.78%]

If "land_area" < 6 236 971.5
And "land_area" >= 4 113 504
And "population" < 2 394.5
And "population" >= 106.5
And "median_income" < 72 500
And "households" >= 30
Then Region = 5 [Confidence = 71.43%]

If "households" < 30
And "land_area" < 21 839 152
Then Region = 5 [Confidence = 64.29%]
```

Figure 4. Description rules of regions described with many rules, obtained using the proposed process.

B. *Spatial Point Clustering*

For this second proof of concept, data obtained from meteorological stations in Argentina on June 6, 2016 were used for this experiment [29]. The considered attributes of the mentioned data are shown in Table 2. The values were normalized.

TABLE II. DESCRIPTION OF THE ATTRIBUTES OF THE METEOROLOGICAL DATA USED IN THE SECOND PROOF OF CONCEPT

Attribute	Description
Lat	Meteorological Station Coordinates
Long	
TMin	Minimum temperature measured on June 6, 2016
TMax	Maximum temperature measured on June 6, 2016
TAv	Average temperature measured on June 6, 2016

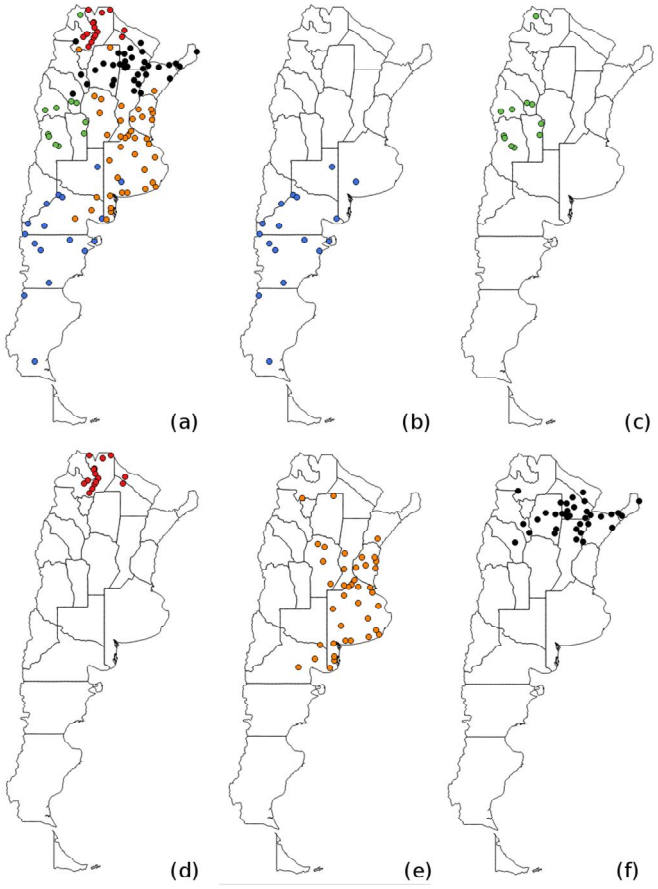


Figure 5. (a) Argentinean meteorological stations distribution with data on June 6, 2016. (b) Meteorological stations in cluster 0 (c) Meteorological stations in cluster 1. (d) Meteorological stations in cluster 2. (e) Meteorological stations in cluster 3. (f) Meteorological stations in cluster 4.

The algorithm selected for the proof of concept was DBSCAN [15], a density-based algorithm for clustering using the implementation available in the data mining software WEKA [30], resulting in 5 spatial clusters with the distribution shown in Figure 5. Later, the clusters were integrated in a

single data file and used as input for Top Down Induction of Decision Trees (TDIDT) algorithm C4.5 [21] implemented in the data mining software Tanagra [28], using the cluster column as target attribute and the non-spatial attributes TMin, TMax, TAv as input attributes, with the non-normalized values of each of them, resulting in a decision tree that yields the rules that can be seen in Figure 6.

```
IF TAv < 9.85
AND TMin > -0.65
AND TMax < 13.5
THEN Cluster 0

IF TAv < 9.85
AND TMin < -0.65
THEN Cluster 1

IF TAv >= 9.85
AND TMin >= 7.2
THEN Cluster 2

IF TAv < 9.85
AND TMin > -0.65
AND TMax >= 13.5
THEN Cluster 3

IF TAv >= 9,85
AND TMin < 7,2
THEN Cluster 4
```

Figure 6. Description rules of the spatial clusters obtained using the proposed process.

The description rules allow differentiating clusters with 83.19% confidence. Cluster 0 has an average temperature lower than 9.85ºC, a minimum temperature higher than 0.65ºC below zero and a maximum temperature lower than 13.5ºC. This last value can distinguish cluster 0 from cluster 3, which has a maximum temperature higher than or equal to 13.5ºC.

On the other hand, cluster 1 can be characterized by its low average temperature and minimum temperature, with values lower than 9.85ºC and -0.65ºC respectively, while the same values are higher than 9.85ºC and 7.2ºC in cluster 2.

Finally, cluster 4 has average temperatures higher than 9.85°C and minimum temperatures lower than 7.2ºC.

In this case, due to the fact that all the attributes of the data have been used for the clustering and not only the spatial attribute, as in the previous proof of concept, the confidence was reduced to 83.19%, as can be seen in Figure 5, where clusters overlap spatially. However, each cluster is homogeneous enough to be described using only one rule for each of them.

V. CONCLUSION

A knowledge discovery process for description of spatially referenced clusters that works with any kind of clusters, i.e. regions, clusters of spatial objects and point patterns, and the description of the characteristic values of the attributes of its members using decision tree learning algorithms is proposed in this paper.

This knowledge discovery process provides a systematic way for business intelligence to obtain relevant information about automatically generated spatial clusters to be used in the decision making, in contrast to existing algorithms. Having a process that specifies the activities to conduct the analysis in a systematic way makes the clustering task more flexible for information mining engineers and decision-makers.

Two proofs of concept of the process using real data have been shown using two different kinds of spatial clusters: the first one uses an algorithm to generate regions, and the second one discovers clusters of spatial points.

In future works, the behavior of the process on data with many non-spatial attributes will be investigated, as well as the influence of choosing different spatial clustering algorithms.

ACKNOWLEDGMENTS

The research presented in this paper was partially funded by the PhD Scholarship Program to reinforce R+D+I areas (2016-2020) of the Technological National University, Research Project 80020160400001LA of National University of Lanús, and PIO CONICET-UNLa 22420160100032CO of National Research Council of Science and Technology (CONICET), Argentina. The authors also want to extend their gratitude to Kevin-Mark Bozell Poudereux for proofreading the translation.

REFERENCES

[1] García-Martínez, Ramón, Paola Britos, and Dario Rodríguez. "Information mining processes based on intelligent systems." International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer Berlin Heidelberg, 2013.

[2] Gopal, Ram, James R. Marsden, and Jan Vanthienen. "Information mining—Reflections on recent advancements and the road ahead in data, text, and media mining." (2011): 727-731.

[3] Mennis, Jeremy, and Diansheng Guo. "Spatial data mining and geographic knowledge discovery—An introduction." Computers, Environment and Urban Systems 33.6 (2009): 403-408.

[4] Kataria, Poonam, and Navpreet Rupal. "Mining Spatial Data & Enhancing Classification Using Bio-Inspired Approaches."International Journal of Science and Research (IJSR). 2012.

[5] Brimicombe, Allan J. "A dual approach to cluster discovery in point event data sets." Computers, environment and urban systems 31.1 (2007): 4-18.

[6] Guo, Diansheng. "Regionalization with dynamically constrained agglomerative clustering and partitioning (REDCAP)." International Journal of Geographical Information Science 22.7 (2008): 801-823.

[7] Yang, Xiankun, and Weihong Cui. "A novel spatial clustering algorithm based on delaunay triangulation." International Conference on Earth Observation Data Processing and Analysis. International Society for Optics and Photonics, 2008.

[8] Zhong, Caiming, Duoqian Miao, and Ruizhi Wang. "A graph-theoretical clustering method based on two rounds of minimum spanning trees." Pattern Recognition 43.3 (2010): 752-766.

[9] Deng, Min, et al. "An adaptive spatial clustering algorithm based on Delaunay triangulation." Computers, Environment and Urban Systems 35.4 (2011): 320-332.

[10] Liu, Qiliang, et al. "A density-based spatial clustering algorithm considering both spatial proximity and attribute similarity." Computers & Geosciences 46 (2012): 296-309.

[11] Shah, Glory H., C. K. Bhensdadia, and Amit P. Ganatra. "An empirical evaluation of density-based clustering techniques." International Journal

of Soft Computing and Engineering (IJSCE) ISSN 22312307 (2012): 216-223.

[12] Nisa, Karlina Khiyarin, Hari Agung Andrianto, and Rahmah Mardhiyyah. "Hotspot clustering using DBSCAN algorithm and Shiny web framework." Advanced Computer Science and Information Systems (ICACSIS), 2014 International Conference on. IEEE, 2014.

[13] Santoso, Aries, and Karlina Khiyarin Nisa. "Cloud Computing Application for Hotspot Clustering Using Recursive Density Based Clustering (RDBC)." IOP Conference Series: Earth and Environmental Science. Vol. 31. No. 1. IOP Publishing, 2016.

[14] Popat, Shraddha K., and M. Emmanuel. "Review and comparative study of clustering techniques." International Journal of Computer Science and Information Technologies 5.1 (2014): 805-12.

[15] Ester, Martin, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." Kdd. Vol. 96. No. 34. 1996.

[16] Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (1998). Density-based clustering in spatial databases: The algorithm gdbscan and its applications. Data mining and knowledge discovery, 2(2), 169-194.

[17] Martins, Sebastian, Patricia Pesado, and Ramón García-Martínez. "Intelligent Systems in Modeling Phase of Information Mining Development Process." International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer International Publishing, 2016.

[18] Martins, Sebastian, Patricia Pesado, and P. García-Martínez. "Information Mining Projects Management Process." Proceedings 28th International Conference on Software Engineering & Knowledge Engineering. Pág. 2016.

[19] Silver, Bruce. "BPMN Method and Style, with BPMN Implementer's Guide: A structured approach for business process modeling and implementation using BPMN 2.0." Cody-Cassidy Press, Aptos, CA 450 (2011).

[20] Hinneburg, Alexander, and Daniel A. Keim. "An efficient approach to clustering in large multimedia databases with noise." KDD. Vol. 98. 1998.

[21] Quinlan, J. Ross. C4. 5: programs for machine learning. Elsevier, 2014.

[22] Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

[23] Ho, Tin Kam. "Random decision forests." Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on. Vol. 1. IEEE, 1995.

[24] Ho, Tin Kam. "The random subspace method for constructing decision forests." IEEE transactions on pattern analysis and machine intelligence 20.8 (1998): 832-844.

[25] Delaunay, Boris. "Sur la sphere vide." Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk 7.793-800 (1934): 1-2.

[26] Complete US City Data – All In One Place. Sample Dataset. Online: https://www.uscitieslist.org/. Consulted on November 28, 2016.

[27] Aurenhammer, Franz. "Voronoi diagrams—a survey of a fundamental geometric data structure." ACM Computing Surveys (CSUR) 23.3 (1991): 345-405.

[28] Rakotomalala, Ricco. "TANAGRA: a free software for research and academic purposes." *Proceedings of EGC*. Vol. 2. 2005.

[29] National Institute of Agricultural Technology (INT A). Daily Data. SIGA – Agrometeorological Information and Management System. Argentina (2016). On-Line: http://siga2.inta.gov.ar/en/datosdiarios/. Consulted on June 6, 2016

[30] Hall, Mark, et al. "The WEKA data mining software: an update." *ACM SIGKDD explorations newsletter* 11.1 (2009): 10-18.

Conceptual Integrity of Software Systems: Architecture, Abstraction and Algebra

Iaakov Exman
Software Engineering Department
The Jerusalem College of Engineering – JCE - Azrieli
Jerusalem, Israel
iaakov@jce.ac.il

***Abstract*— Conceptual Integrity has been claimed to be the essence of high-quality software system design. On the other hand, it has been a rather elusive attribute of software systems, challenging various attempts of a clear-cut characterization. This paper evolves in this direction by two means: first, by analysis and clarification of open issues in architecture and abstraction terms; second, by pointing out to a mathematical formulation in algebraic terms. This paper also serves as a broad introduction to discussions on "Conceptual Integrity of Software Systems".**

Keywords: Conceptual Integrity; software system design; architecture; abstraction; algebra; Linear Software Models; Modularity Matrix; Conceptual lattice; plausibility criteria; design constraints.

I. INTRODUCTION

Frederick Brooks [2], [3], based upon his extensive experience with system development, in particular the first families of OS/360 operating systems, proposed that Conceptual Integrity is essential for high-quality software system design. It takes some time to assimilate this idea, but even after reading about it once and again and having second and third thoughts, Conceptual Integrity remains attractive, but a quite elusive notion.

The first task of this paper is to introduce the notion, its attractiveness and why it is still elusive. Then, we argue in favor of mathematical formalization, like in any respectable science. The overall purpose of this paper is to try to open and discuss deep issues on "Conceptual Integrity of Software Systems" and to point out to a formal solution to these issues.

A. The Notion of Conceptual Integrity

Software system design and development, as it is common practice nowadays, gives a superficial impression of unrestricted flexibility, where everything is allowed and nothing is forbidden. Often practitioners think that software design and development is an art, rather than a science. People exposing this opinion may express distaste for suggestions of mathematical description of the processes involved.

The feeling of unrestricted flexibility and its artistic connotation is acquired from the first experiences with computer programs that one writes. Except for an apparent arbitrariness of the programming language syntax, one has full confidence in the choice of the preferred programming construct – this is the unrestricted flexibility – and the program is supposed to run for sure. Then, happens the inevitable bug, demanding a subtle correction – this is the artistic connotation.

A widespread rational response to the *artistic flexibility* is to introduce methodologies. If one is methodical, then the price of software design and development should decrease. This is the first encounter with elusiveness (cf. the word "Mythical" in the title of the earlier book by Brooks). Any experienced software developer knows that methodologies are not the panacea that they promise to be. They fail, sometimes miserably.

Then, Conceptual Integrity, by Brooks, enters the stage. This is a deeper response to the software system development problem. We explain the notion in literally *architecture* terms, exactly as in Brooks' books, while presenting open issues in section III.

B. Paper Organization

The remaining of the paper is organized as follows. Section II refers to related work. Section III introduces open issues of Conceptual Integrity in a broad context. Section IV focuses on architecture principles. Section V turns to abstraction principles. Section VI summarizes algebraic principles of software design. Section VII concludes with a discussion.

II. RELATED WORK

A. Conceptual Integrity Outside Software Systems

This paper refers to Architecture of buildings in general, following Brooks' metaphors used in his books to illustrate notions of Conceptual Integrity. In this context, we refer here to a few architecture-related modeling techniques and ontologies.

DeLuca and co-authors [5] describe a generic formalism for semantic modeling of architectural elements, which compose buildings of historic interest in classical architecture. Doerr [8] reviews ontologies for cultural heritage; he emphasizes physical objects in archeology, including architecture. Quattrini et al. [22] describe modern computer-based techniques for semantically-aware 3D modeling of architecture.

DOI: 10.18293/SEKE2017-206

B. *Conceptual Integrity Within Software Systems*

Conceptual Integrity is closely related to efforts regarding various development phases of software systems. These efforts are among others: requirements engineering, conceptual modeling, integrity verification, software system design, and software architecture planning. The relevant literature is very extensive, and we provide just a few representative pointers.

Jackson and co-authors [6],[17],[18] analyzed widely used software systems, such as Git, in terms of Conceptual Integrity, suggesting design improvements.

Insfran et al. [16] refer to conceptual modeling based on requirements engineering. For these authors conceptual modeling is an UML object-oriented approach rather than based on conceptual integrity. Nevertheless, there are similarities between these approaches, as discussed later on.

Cabot and Teniente [4] refer to integrity checking of UML/OCL conceptual schemas. Integrity here means software state conditions that must be satisfied. Sometimes it specifically refers to avoidance of critical system malfunction. Conceptual schemas are basic relations between concepts within the general knowledge required by any information system (see also e.g. the book by Olive [21]).

Kazman and Carriere [19] reconstruct a software system architecture using *conceptual integrity* as a guideline. Their goal is to achieve a restricted number of components connected in regular ways, with internally consistent functionality.

C. *Mathematical Conceptual Integrity of Software*

In this work we shall mainly refer to the Modularity Matrix [9],[10],[13] which is based upon linear algebra. Other matrices have been used for modular design. For instance, the Design Structure Matrix (DSM) is an integral part of the 'Design Rules' by Baldwin and Clark [1]. It has been applied for various kinds of systems, including software systems. DSM design quality is estimated by an external economic theory superimposed on the DSM matrix.

Conceptual lattices, analyzed within Formal Concept Analysis (FCA) were introduced in Wille [24]. A generic review of its mathematical foundations is given by Ganter and Wille [14]. Conceptual Lattices have been shown to be equivalent to Modularity Matrices (e.g. Exman and Speicher [12]), linking the algebraic characteristics to conceptual ones.

III. OPEN ISSUES: ARCHITECTURE, ABSTRACTION, ALGEBRA

We now consider the open issues of Conceptual Integrity in a wide context. We start by referring to it in architecture terms. Then we consider abstraction and algebraic formulations.

A. *Open Issues: Architecture*

In Brooks' book "The Mythical Man-Month" [2] in front of the fourth chapter opening (page 41) there is a photo of the interior of the Reims cathedral, planned by Jean D'Orbais. An annotated sketch of the cathedral is seen in Fig. 1 in this paper. It has an imposing huge height relative to humans, as usual for medieval gothic cathedrals, implying the difficulty to actually build it.

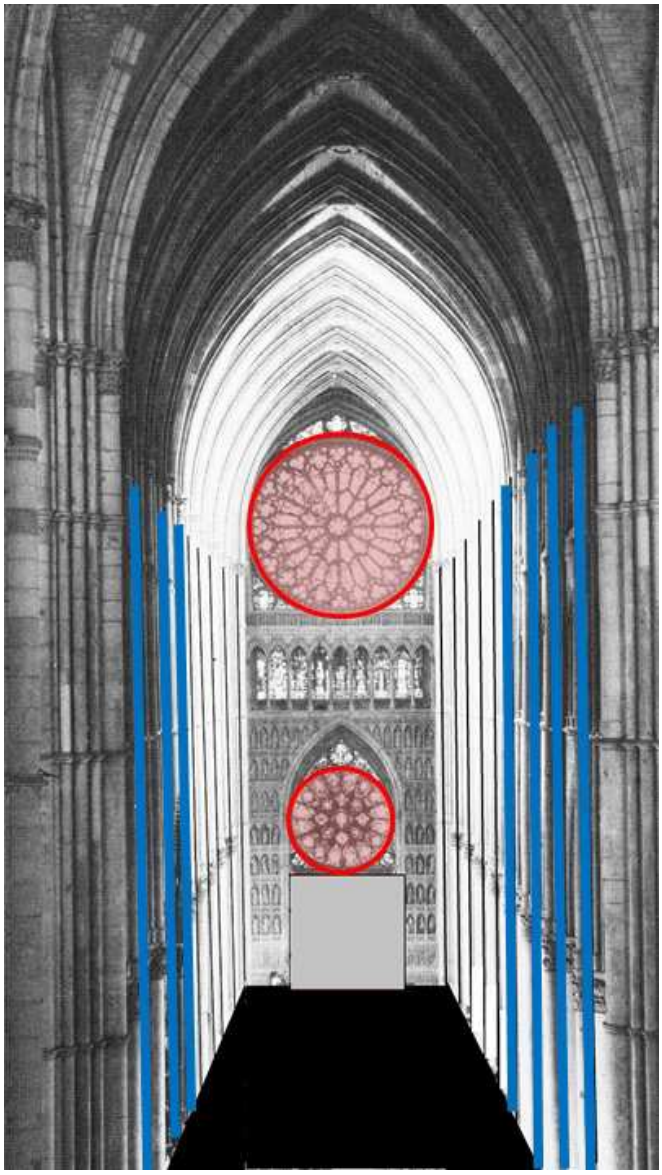


Figure 1. Reims Cathedral Interior: sketchy annotations – one can see the perfect symmetry of its elements: two rows of very high parallel columns (annotated by vertical blue color line segments), two stained glass rose windows (annotated by a pink background within a red circle), and above the columns the parallel gothic arches.

Nonetheless, despite the incredible weight of its stones, it displays elegance, coherence, and symmetry of its component forms: e.g. long repetitive rows of identical very high columns and two symmetrical stained glass rose windows above the altar region.

Another example is the renaissance cathedral of Firenze, whose symmetric dome with repetitive elements was designed by Brunelleschi. It is seen in Fig. 6-4 (page 75) in "The Design of Design" [3] book by Brooks.

Thus, Conceptual Integrity in classical, medieval and renaissance *architecture* means that the overall structure is immediately recognized by its repetitive and symmetric components' consistency, and is very attractive by its esthetics.

Figure 2. Bilbao Museum by Frank Gehry: sketchy annotations – one perceives the intentional total absence of symmetry of its elements. It displays one very high column (annotated by two vertical blue linear segment bounds) in its entrance; compare its height with that of human visitors, which are pointed out by the (dark blue) arrow. The overall asymmetric structure should resemble a ship contour (annotated by red linear segments, of various sizes and non-parallel directions). The outer metal structures reflect light by fish-like scales.

The Guggenheim Museum in Bilbao, designed by the architect Frank Gehry, has been metaphorically referred to as a modern cathedral, due to its enormous size, in particular the huge height of its atrium [15]. Fig. 2 is an annotated sketch of it. Moreover, it seems to be very consistent, by the similarity of materials and forms, to resemble a ship – since Bilbao is a port. In fact, it was designed with the support of the Digital Project (see e.g. [7]), which itself is based upon CATIA a sophisticated software system used to plan modern aircraft.

On the other hand, nothing is repetitive or symmetric in the Bilbao Museum. There are no two identical components in the Museum. We thought that we had the keys to the notion of Integrity, but, modern *architecture* breaks down the older keys. This is our second encounter with elusiveness. We are left with the open issue:

Open Issue #1 – Conceptual Integrity in Architecture
Despite symmetries being fundamental to physics – i.e. they correspond to conservation principles – and being important in classical architecture, symmetries are not ubiquitous in modern architecture.
What is the generalization of symmetry that is common to Conceptual Integrity in all kinds of architecture?

Paradoxically one still could say that the Bilbao Museum displays an internal Conceptual Integrity – the ship outline – and even to the apparently dissimilar buildings of the city of Bilbao. Moreover, the outer titanium metal sub-structures reflect light like fish scales! But how is ship and fish related concepts? This is discussed in the next sub-section on Abstraction.

B. Open Issues: Abstraction

We shall refer to abstraction in two senses: first, geometric or image abstraction; second, conceptual abstraction.

Regarding geometric abstraction, we mean that any considerations of Integrity are not taken with respect to the actual buildings referred to in the previous architecture sub-section. Neither real stone blocks, nor titanium metal surfaces are necessary to perceive symmetry or the "ship" shape.

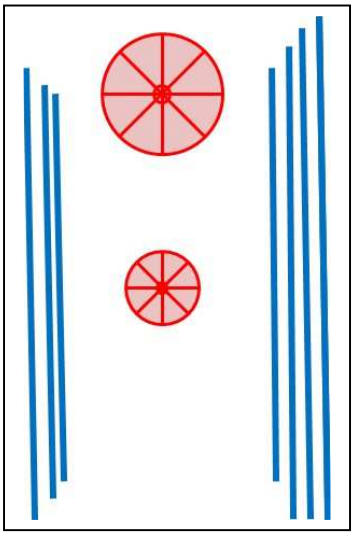


Figure 3. Reims Cathedral Geometric Abstraction: only sketchy annotations – this figure abstracts Fig. 1 to just the schematic representation of the rows of parallel columns (in blue) and the two stained glass rose windows (in red).

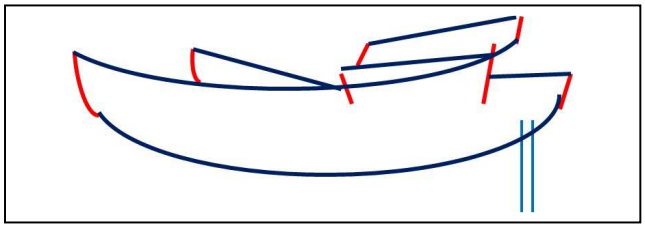


Figure 4. Bilbao Museum Geometric Abstraction: only sketchy annotations – this figure abstracts Fig. 2 to just the schematic representation of the exactly copied contour segments (in red) and the column bounds two vertical (light blue) segments. The contour segments were in addition linked by round (dark blue) lines, to facilitate our perception of the overall "ship" form.

One performs quite complex abstraction operations in the human brain, which are simulated by computer image processing – noise elimination, segmentation, and object recognition – to extract just the lines needed to infer either symmetry as in Fig. 3 or a "ship" shape as in Fig. 4.

Conceptual abstraction is a further step beyond geometric abstraction. In the cathedral case the relevant concepts are "*columns*" and "*stained glass rose window*", together with the even more abstract concept of "*symmetry*". The next step in conceptual abstraction is the usage of architecture domain ontologies (see e.g. [8]). In the Bilbao Museum case, relevant concepts could be "*port*" of Bilbao, "*ship*" and "*fish*". These could be found in Maritime domain ontologies (see e.g. [23]). But, the conceptual jump from these specific concepts to a Museum of modern art architecture is far from obvious. Then:

Open Issue #2 – Abstraction Conceptual Integrity
Abstraction Conceptual Integrity seems to be intrinsically associative irrespective of the relevant domain.
How to formally capture the associative character of abstraction Conceptual Integrity?

C. Open Issues: Algebra

Let us temporarily take for granted that Algebra is the mathematical basis of our formal approach to Conceptual Integrity – an approach extensively justified elsewhere (see e.g. [10], [11]), with respective arguments presented later on in this paper. In order to apply algebra to concepts we do not try to get an answer to Open Issue #2. Instead, we assume that the associative process of choosing sets of abstraction concepts – as discussed in the previous sub-section – is done by human software engineers, without any current attempt to formalize this process.

The algebraic phase of Conceptual Integrity starts with the chosen sets of concepts that humans find necessary to build a desired software system. We have called the specific ontology relevant to a desired software system as "*application ontology*". One can easily see that application ontologies have a striking similarity to UML class diagrams. Ontology concepts correspond to classes, and relationships among classes are of three kinds: inheritance (a sub-class is a type of class), composition (wholes are made of parts) and association (say usage of the functionality of another class). Thus, we deal with classes and their functionalities, and their respective generalizations.

What is lacking in the above picture? The answer is: we lack criteria to verify that the choice of concepts is reasonable. This is the role of algebra. Thus:

Open Issue #3 – Algebraic Conceptual Integrity
Assume that humans choose the needed concepts relevant to a desired software system, by a black-box algorithm.
What are the criteria to verify that the concepts choice indeed comply with an idea of Conceptual Integrity for the desired software system?

IV. Architecture Principles

The answer to the Open Issue #3 starts with the understanding that in any domain there are constraints limiting solutions. We start as in the previous section with principles of architecture.

A. Structure: Buildings and Software

The basic constraints obeyed by any building, either in antiquity or in modern times are first and foremost the laws of static – the branch of classical mechanics (itself a field of physics) dealing with structures which have no motion. One should have columns and beams to support structures of a building, a bridge over a bay or tunnels below a river. These columns and beams have a material composition, say concrete or steel, and suitable sizes and forms. Modern constructions have additional constraints, such as heating, acoustics, distribution of electricity and water tubing.

Software structures, just as buildings, are hierarchical with classes and patterns – as sets of classes – being the relevant structural units. The relevant constraints are enforced by the above mentioned relationships already found in the application ontologies – viz. inheritance, composition and association.

B. Behavior: Flying and Running Systems

Physical systems displaying motion, such as airplanes, autonomous cars, robots, helicopters, boats and drones (unmanned aircraft systems) may fly, walk, navigate, etc. In addition to structural constraints, they have kinematic constraints limiting their motion in terms of speed and possible trajectories.

Software systems when running also have constraints on their speeds, communication and memory usage. These are certainly affected by their structural constraints. With the increasing usage of autonomous systems, with embedded software, the software itself is also affected by the constraints of the containing physical systems.

C. Modularity

Modularity is an important consideration for any system, as it facilitates development, building and understanding of systems. Modularity is achieved by simplistic repetitive reuse of the same units again and again – such as the stones in the columns of the Reims cathedral, and wheels in mobile vehicles.

Modularity maybe also achieved by more sophisticated means than strict repetition. The titanium metallic surfaces of the Bilbao Museum are each of them unique, but they are generated by a software system which reuses the same technology to obtain different shapes.

Modularity is the result of complying with constraints, partially relaxing them as far as possible while minimizing undesirable effects. This is true in particular for software systems.

V. Abstraction Principles

In this section we deal with abstraction principles. Abstraction here is understood as conceptual abstraction for system design. We further focus only on embedded software or purely software system design.

The abstraction principles express the necessary constraints which limit design to modular solutions. Concomitantly these provide the criteria to verify that design solutions are optimal under these constraints. In terms of the abstract concepts, this is the meaning of *Conceptual Integrity*.

Conceptual abstraction principles were first formulated by Brooks in his books [3]. Here we succinctly review these principles. Note that all the three principles are formulated as ***negative*** expressions, i.e. "**Do not…**", which are effectively constraining design and meaning.

A. Abstract Propriety

Brooks concisely formulates propriety as: "***Do not introduce what is immaterial***".

He explains this principle by an example of propriety in computers, viz. the representation of zero in twos-complement notation, which obeys the constraint of not attaching a sign to zero.

In contrast, signed-magnitude and ones-complement representations do attach a sign to zero, which is extraneous and inconsistent with the original meaning of the "zero" concept. The consequence of these representations is addition

of artificial rules needed to characterize the behavior of zero in arithmetic operations.

B. Abstract Orthogonality

Brooks concisely formulates orthogonality as: "***Do not link what is independent***".

He explains this principle by a basic example of orthogonality in computers, viz. the operation of a (software or embedded) alarm clock. Two functionalities of such a clock are lighting and alarm. Orthogonality means that these two functionalities obey the constraint of actually being independent.

In contrast, if the alarm would operate only when the clock is illuminated, it would violate orthogonality, since one is artificially linking two unrelated functionalities.

C. Abstract Generality

Brooks concisely formulates generality as: "***Do not restrict what is inherent***".

He explains this principle by a surprising example of generality of an operation in a processor, viz. the operation "*restart*". Its original purpose was to restart a process after an interruption. But the design generality enabled its use as returning from a subroutine. The obeyed constraint here is avoiding incidental restriction.

In contrast, if it were strictly allowed only for the original purpose, one would clutter the design by introducing another almost overlapping concept for the second kind of usage.

VI. Algebra: Principles of Software Design

The power of a mathematical formalism is to express the abstract constraints in a precise way, enabling calculations upon the representation of a given software system design – e.g. of eigenvectors [11],[20] to obtain software modules –, and the verification whether the design comply with the imposed constraints. Moreover, one can improve the software design, based upon the verification results.

The choice of linear algebra structures, such as the Modularity Matrix and its equivalents, is justified by the following reasons:

- ***Expressivity*** – algebraic structures are expressive enough to represent the wide variability of software structures;
- ***Constraints Nature*** – it is very natural to formulate Brooks' abstraction constraints in algebraic terms, as will be seen in the following sub-sections;
- ***Conceptual Meaning*** – since each Modularity Matrix of a software system is equivalent to its corresponding Conceptual Lattice, the Conceptual meanings are immediately taken into account.

A. Algebraic Propriety

The algebraic translation of Brooks' Propriety is based upon the following ideas [10]. Instead of sets of classes (the concepts), one works with vectors of classes. Structors are generalized classes to all levels of the hierarchical software system. Structors provide functionalities. Thus, one also has functional vectors.

In order to avoid immaterial additions, i.e. the design goal is to minimize the representation of a software system, the algebraic ***Propriety*** constraint is: ***linear independence*** of all the structor vectors and of all the functional vectors in the Modularity Matrix of the given software system. Any dependent vector is superfluous and discarded.

B. Algebraic Orthogonality

The algebraic translation of Brooks' Orthogonality is even more immediate. One literally uses exactly the same word orthogonality, but now with the exact algebraic meaning of "zero scalar product of a pair of vectors".

The algebraic ***Orthogonality*** constraint is: ***orthogonality*** of all structor vectors and all functional vectors inside a module to the respective vectors in all other modules in the Modularity Matrix of the given software system. Thus, different software modules actually deal with different concerns, literally having different conceptual meanings.

C. Algebraic Generality

The algebraic translation of Brooks' Generality is that since the definition of a structor or its functionalities are not strictly repeated in another location of the same software system, one still can re-use these structors and their functionalities elsewhere in the same system by means of composition or by means of aspect-oriented design.

The algebraic ***Generality*** constraint is: ***generality*** once again minimizes the number of appearances of structor vectors and functional vectors in various hierarchical levels of the Modularity Matrix of the given software system. On the other hand, in an aspect-oriented design fashion, one puts the general structors/functionals in a high enough hierarchical level, to be accessible from anywhere in the system.

VII. Discussion

We here summarize the important points of this paper. Brooks' used architecture of cathedrals to justify his Conceptual Integrity principles. Indeed the architecture of buildings provides fruitful metaphors in our context, as we have seen that one obtains valuable concepts from the building abstractions. But, even more importantly, they lead to open issues challenging simplistic notions of Conceptual Integrity.

A. Eliciting Concepts

Concept elicitation from system stakeholders, using system sketches, drafts or early models, is an activity related to requirements engineering. It demands high human capabilities and in this paper this activity was left to the human engineers. We currently give up any effort to mathematical formalize this activity. It may well be that we shall return to this basic issue in future work.

We deliberately assumed that one starts the Conceptual Integrity analysis from an initial list of concepts – translated

into structor and functional vectors. This initial list may change along the design and development processes.

B. *Criteria for Integrity*

Brooks' principles actually are constraints on the software system design solutions. Once this is understood, one can clearly express algebraic equivalent formulations of the same principles.

The propriety and orthogonality constraints received a very natural translation to linear algebra notions. The generality principle seems to deserve further investigation.

C. *Main Contribution*

The main contribution of this paper is a set of open issues to be discussed within a Conceptual Integrity forum, and the clear understanding of its principles, as mathematical constraints on the design solution for a software system.

REFERENCES

[1] C.Y. Baldwin and K.B. Clark, *Design Rules*, Vol. I. The Power of Modularity, MIT Press, Cambridge, MA, USA, 2000.

[2] F.P. Brooks, *The Mythical Man-Month – Essays in Software Engineering* – Anniversary Edition, Addison-Wesley, Boston, MA, USA, 1995.

[3] F.P. Brooks, *The Design of Design: Essays from a Computer Scientist*, Addison-Wesley, Boston, MA, USA, 2010.

[4] J. Cabot and E. Teniente, "Incremental Integrity Checking of UML/OCL Conceptual Schemas", J. Systems and Software, Vol. 82, pp. 1459-1478, Sept. 2009. DOI: http://doi.org/10.1016/j.jss.2009.03.009

[5] L. De Luca, P. Veron and M. Florenzano, "A generic formalism for the semantic modeling and representation of architectural elements", Visual Computer, Feb. 2007. DOI: http://dx.doi.org/10.1007/s00371-006-0092-5

[6] 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.

[7] Digital Project, https://en.wikipedia.org/wiki/Digital_Project

[8] M. Doerr, "Ontologies for Cultural Heritage", in Staab and Studer (eds.) Handbook on Ontologies, Springer-Verlag, Berlin, 2009, DOI: http://dx/doi/org/10.1007/978-3-540-92673-3

[9] I. Exman, "Linear Software Models", Proc. 1st SEMAT Workshop on a General Theory of Software Engineering, KTH Royal Institute of Technology, Stockholm, Sweden, 2012. http://semat.org/wp-content/uploads/2012/10/GTSE_2012_Proceedings.pdf. video of this paper in the web site: http://www.youtube.com/watch?v=EJfzArH8-ls.

[10] 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.

[11] 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: http://dx.doi.org/10.1142/S0218194015500308

[12] I. Exman and D. Speicher, "Linear Software Models: Equivalence of the Modularity Matrix to its Modularity Lattice", in Proc. 10th ICSOFT'2015 Int. Conference on Software Technology, pp. 109-116, ScitePress, Portugal, 2015. DOI: 10.5220/0005557701090116

[13] I. Exman, "Linear Software Models: An Algebraic Theory of Software Composition", in Proc. 28th Int. Conf. on Software Engineering and Knowledge Engineering, Keynote Abstract, KSI Research, Redwood City, CA, USA, 2016.

[14] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*, Springer-Verlag, Berlin, Germany, 1998.

[15] Guggenheim Museum in Bilbao by Frank Gehry, https://en.wikipedia.org/wiki/Guggenheim_Museum_Bilbao

[16] E. Insfran, O. Pastor and R. Wieringa, "Requirements Engineering-Based Conceptual Modeling", Requirements Eng. Vol. 7, pp. 61-72, June 2002, DOI: http://dx/doi/org/10.1007/s007660200005

[17] 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

[18] 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.

[19] 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.

[20] U. von Luxburg, "A Tutorial on Spectral Clustering", *Statistics and Computing*, 17 (4), pp. 395-416, 2007. DOI: 10.1007/s11222-007-9033-z

[21] A. Olive, *Conceptual Modeling of Information Systems*, Springer-Verlag, Berlin, 2007.

[22] R. Quattrini, E.S. Malinverni, P. Clini, R. Nespeca and E. Orlietti, "From TLS to HBIM. High Quality Semantically-aware 3D Modeling of Complex Architecture", in 3D Virtual Reconstruction and Visualization of Complex Architectures, pp. 367-374, Avila, Spain, Feb. 2015. DOI: http://dx/doi/org/10.5194/isprsarchives-XL-5-W4-367-2015

[23] G. Santipantakis, K.I.Kotis and G.A. Vouros, "Ontology-Based Data Integration for Event Recognition in the Maritime Domain", WIMS '15, July 2015, Larnaca, Cyprus, DOI: http://dx.doi.org/10.1145/2797115.2797133

[24] 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.

Conceptual Integrity without Concepts

Giorgio Grasso
Department of Cognitive Science
University of Messina
gmgrasso@unime.it

Alice Plebe
Department of Mathematics and Computer Science
University of Catania
alice.plebe@gmail.com

Abstract

It is commonly asserted that one of the most crucial factors in software design is the adoption of consistent and appropriate concepts. This widespread assumption, underlying several researches in software engineering, tacitly postulates the very existence of concepts. What, then, if concepts do not exist? This is the question asked in this paper, where the articulation of possible answers is attempted. As weird as it might sound, doubts on the existence of concepts have been recently cast by few distinguished philosophers of mind, with compelling arguments. One such argument is the heterogeneity of cognitive assets that can be referred to a single concept, and we show that it might be the case for design concepts in the example of Blender, a 3D computer graphics software.

1. Introduction

There is a widely perceived view that there exists something that we call "concept". Concepts are allegedly used to categorize the world, to store knowledge, for reasoning, and for drawing analogies. This is certainly the view held by Frederick Brooks when he famously stated that the success of software system design rely heavily on the precise identification and preservation of "concepts" [4]. It is reasonable to assume that the integrity of design concepts throughout software development was based on the implicit postulate that concepts are objective and stable entities. We will show that in fact this is not always a necessary requirement, but it is certailny usuallly assumed.

This tacit confidence on the ontology of concepts has surrounded all the research domain in conceptual design for software engineering that followed Brooks' ideas [9, 2, 8]. Moreover, even the precise nature of concepts, with very few exception [16], has never been inquired.

Yet, the ontological status of concepts is highly controversial, and, recently, compelling arguments have been proposed to challenge the very existence of something we call "concepts" [20, 21, 25]. The difficulties in finding a suitable ontological status for concepts, and the reasons for eventually denying any such status, will be briefly reviewed in §2.

While not entering into this debate as such, this paper questions how the issue of conceptual integrity in software design will fare, if it turns out that concepts in fact do not exist. One of the main troubles affecting concepts, as traditionally conceived, is their heterogeneity, that will be discussed in §2.1. If a linguistic label used to refer to a specific design concept actually corresponds to a plurality of heterogeneous cognitive assets, then, arguably, a serious threat undermines the whole idea of conceptual integrity. Scholars arguing for the elimination of concepts have also proposed new theoretical entities, more congruent with the current knowledge of human cognition. One, here discussed in §2.2, is that of "unicept", a sort of concept unique for a human agent. The trouble for conceptual integrity can be now traced to the simple fact that, unlike concepts, unicepts are not objective entities, thus cannot be generalized across individuals.

In §3 we will show that it may be the case for Blender, a popular open-source software for 3D computer graphics. This software is developed under a wide collaborative basis, and it is built on to a set of firm design concepts concerning its user interface. Yet, Blender is often not regarded as easy to use, even by experienced graphics developers familiar with other software applications. We argue this is a case of heterogeneous cognitive assets of, apparently, same concepts, nothing exceptional if concepts do not exist.

However, we argue that the lack of a well-founded ontology for concepts does not undermine completely the need and practice of conceptual integrity, as will be demonstrated in the case of the software Blender. Rather, it requires awareness that design concepts are not necessarily carved on stone tablets just because they are expressed in words, therefore the ideas of software architects may not coincide with those expected by implementers or users.

DOI reference number: 10.18293/SEKE2017-203

2. Do concepts exist?

Doubts about the existence of concepts were raised by the extreme difficulties found in every attempts to define exactly what concepts are, and how they work.

In modern philosophy extensive work on the nature and origin of concepts was carried by the English empiricists, especially John Locke [19] and David Hume [15]. In their idiom concepts were called "ideas", similarly to Frederick Brooks who, in his books, uses "design ideas" as a perfect equivalent of "design concepts". Ideas, for Locke and Hume, are perceptually based, and reason gets its contents from the senses. The empiricist position has been often refuted in the history of philosophy, but recently proposed in a modernized account supported by cognitive and neurocognitive evidences [29].

A large body of work on concepts was carried out by psychologists in the second half of last century, but marginally concerned with a precise characterizations of what concepts are. Concepts are often conflated with categories [23], knowledge [1], and representations [33].

In most contemporary philosophy of mind and philosophy of language "concepts" are used as a non technical equivalent for "propositional attitudes" [27, 10, 22, 29], i.e. propositions that, like beliefs, can be true or false, or, like desires, can be satisfied or unsatisfied. Most of the times philosophers content themselves with this account of concept, and their effort is in explaining how concepts are individuated. For instance, they are supposed to explain what distinguishes our capacity to have propositional attitudes about keyboards as such from our capacity to have propositional attitudes about hard-disks as such. Even if the focus of philosophy and psychology on concepts is different, in the last decades the discussions become entrenched, with philosophers evaluating the virtues of psychological theories of concepts with respect to their own criteria [10, 24, 29].

2.1. (Too) Many concepts of "concept"

Details of the countless theories of concepts in psychology and philosophy are out of the scope of this work, but the diversity of existing theories is at the very core of concerns related to the existence of concepts. For sure, both philosophers and psychologists have always acknowledged the wild differences between concepts and between kinds of concepts, but the received view is that, over and beyond the differences between concepts and between kinds of concepts, there is a set of common relevant properties [26, 13]. Few have proposed that the apparent divergence between concepts is due to a pluralism in scope, for example few properties could be shared by cats and computer keyboards because biological kinds and artifacts have little in common [18]. Others have argued for a pluralism of competences, different kinds of concept are involved in different cognitive competences [28], for example a concept of keyboard used when we categorize visually an object as keyboard could have little in common with a concept of keyboard when reasoning about developing the software driver module for a keyboard.

A different line is held by Machery with his "heterogeneity hypothesis" [21], the proposal that most categories of physical objects, most types of event, and most substances are represented by several concepts that belong to kinds that have little in common. There are at least four different theoretical entities that have been conceived as theories of concepts, and they have little in common.

The classical theory states that a concept is the set of properties which are separately necessary and jointly sufficient for an object to belong to the class of that concept [32], this theory has been largely endorsed before the end the 1960s [17]. Notably, the classical theory had profound influence on computer science, it has been assumed as the foundation for the domain of formal concept analysis. One of the most favored mathematization of concepts in computer science is by sets of objects and attributes, that define, respectively, the extension and the intention of a concept, and the interdependence of concepts is formalized by mathematical lattice structures [12]. This formalization of concepts has proven fruitful in several applications, including text retrieval and mining [7], support in software engineering for identification of object-oriented structures or refactoring activities [34]. While formal concept analysis is still today an active domain of research, his underlying classical theory of concepts has been mostly abandoned in philosophy for decades. It was replaced in the 1970s by the *prototype* paradigm, assuming that we have in mind some properties that are believed to be typically possessed by the members of a class [31, 14]. Few years later a different paradigm of concepts was proposed, that of *exemplar* [5], the idea is that concepts are sets of exemplars, which are bodies of properties believed to be possessed by few particular members of a class. The fourth theory is the *theory* paradigm [6, 30], proposing that concepts are some kind of folk theories, knowledge that can explain the properties of category members. Machery compellingly illustrated how all those theoretical paradigms of concept have little in common, from both the point of view of the types of knowledge they encode, and the kinds of cognitive process they involve.

2.2. Unicepts

The elimination of concepts for Machery is a consequence of the perplexing heterogeneity of theoretical construes, reviewed above, which in fact denote heterogeneous

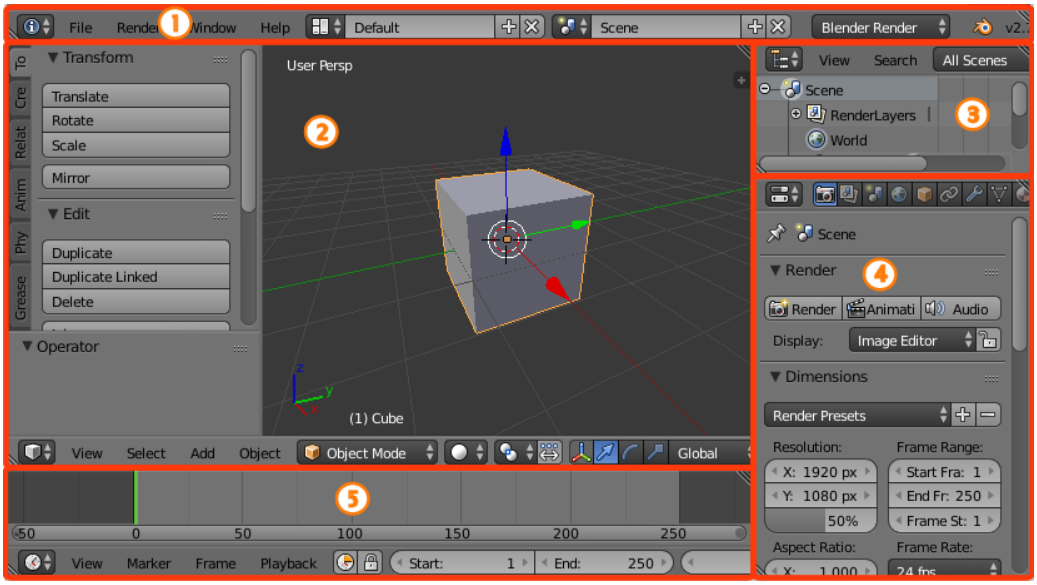


Figure 1. The Blender interface, with highlighted in red five "Editors": Info (1), 3D View (2), Outliner (3), Properties (4) and Timeline (5).

ways in which the mental/neural vehicles hold information about things and properties of things. It is this heterogeneity that led Millikan to introduce a new notion different from "concept", as the fundamental units of cognition: the "unicept". In her words:

> "Uni" is for one, of course, and "cept" is from Latin *capera*, to take or to hold. One's unicept for an object, or property, or kind, or relation etc., takes in many proximal stimulations and holds them as one distal entity. A developed unicept reaches through a radical diversity of sensory impressions to find the same distal thing again. [...] A unicept is a specific individual *faculty* developed for a very specific purpose, the purpose of collecting and integrating information about some particular thing. [25, p.16]

This new theoretical entity differs from "concept" in several ways, which are not essential in this paper. What count here is that, unlike the common view on concepts, unicepts are not entities that people share. Every individual has his/her own private stock of unicepts. In most cases unicepts of different people succeed in gathering information about exactly the same things in the world, but they do this in different ways, sometimes using overlapping cognitive strategies, but also many that are distinct. Moreover, having a unicept involves having a certain kind of ability or capacity to deal, successfully, with an aspect of the world, and there is no genuine unicept unless what it pulls information about is indeed a precise entity in the world. Otherwise, in the words of Millikan, "if it pulls together information about many things, using this as though it were about one thing, then [...] it is an *empty* unicept or at best an *equivocal* unicept.

3. Unicepts in Blender

In order to illustrate cases where design concepts are much better defined in terms of Millikan's unicepts, rather than traditional, universal, concepts, we analyize the user interface of Blender (`www.blender.org`), a widely used open-source computer graphics software [3]. Originally developed in 1995 by Ton Roosendaal as an in-house tool, in 2002 turned into an open source project, that since 2007 has been coordinated by the Blender Institute in Amsterdam, under Roosendaal direction.

Elements in the interface of Blender are organized in strict hierarchical manner, based on the following "concepts":

Window ⇒
Screen ⇒
Areas ⇒
Editors ⇒
Regions ⇒
Tabs ⇒
Panels ⇒
Controls

"Screens" are window layouts, Blender offers a set of pre-defined possible layouts, but it is possible to customize

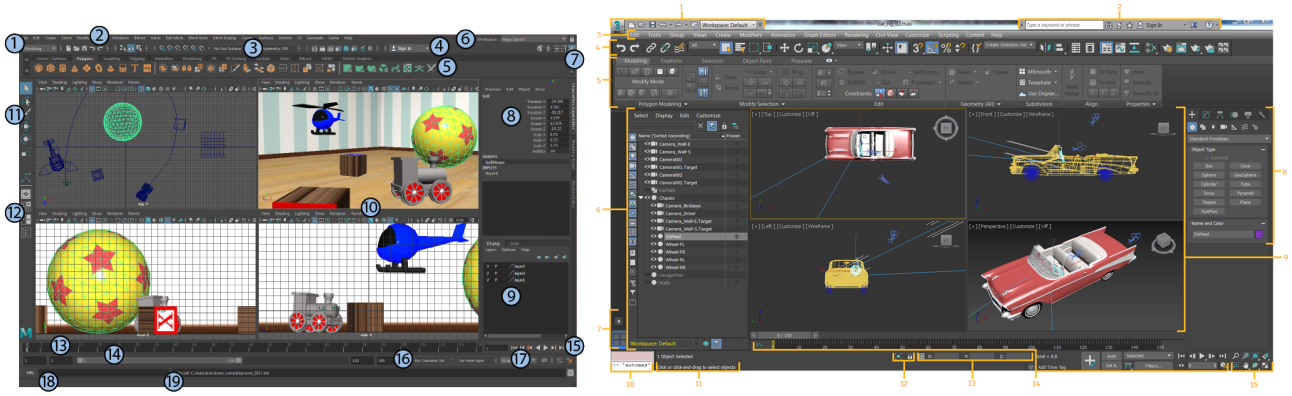


Figure 2. Comparison with interfaces of other 3D computer graphics software: Maya on the left, 3Ds Max on the right.

any of them. Screens are divided up into a number of re-sizable "Areas", each of which containing a particular type of "Editor", like a 3D View Editor (the main interaction with the 3D scene), an Outliner (the overview of scene graphs hierarchy), a Graph Editor (for keyframe interpolation), a Node Editor (node-based shading and compositing tools), and so on. In Fig. 1 a screenshot of the "Window" is given, with a layout of 5 "Areas" hosting different "Editors". "Editors" in turn are split into "Regions", portions of the space inside the "Area" containing smaller structuring elements like tabs and panels with buttons, controls and widgets placed within them. Blender uses a peculiar screen-splitting approach to arrange areas: the "Window" can be divided into an arbitrary number of smaller adjacent "Areas", and each "Area" is always fully visible, and it is straightforward to work in one area and hop over to work in another.

3.1. What "Editor" means

Most of the words here quoted to address the main "concepts" of Blender's interface are of common usage among professional sofware, and not only in the computer graphics domain. The point is that several of this words, even if including properties shared by other software, are characterized by aspects of meaning, and usage, which are distinctive in Blender. For sexample, in the two main alternative to Blender, Maya and 3Ds Max (see Fig. 2) there is a sharp distinction between "editors" and other tools like toolbox shelves (Maya) or ribbons (3Ds Max). In Blender, every instance of an "Area" contains an "Editor", words that, arguably, refers to an unicept, kind of concept held by accustomed users of Blender, and by its developers, but diverges considerably from the set of properties and features other users include in their unicept of "editor".

It is noteworthy that despite this deviation of several concepts from their account in other software, Blender achieved in its user interface a remarkable integrity. There is a high consistency of interaction across all "Editors", with several non-modal tools, that can be accessed efficiently without taking time to select them according to the type of editing. Inside every "Editor" there is a consistent and predictable usage of mouse and keyboard actions for interaction.

For example, most single-key commands have their reverse action with the same key pressed together with the `Alt` modifier. The key `A` means the selection of "All", where this quantifier can refer to a variety of entities depending on the "Editor", like 3D meshes in the case of 3D View, keyframes in the case of Graph Editor, nodes in the Node Editor, and so on. These features are among the most appreciated in the community of Blender users, who find the working efficiency offered by its interface unsurpassed.

Therfore, it is possible to state that Blender, in its user interface, is a good example of, say, *Uniceptual Integrity*. This integrity, which is certainly an important reason of its success, comes at a price. Unicepts are not universally shared, and users that formed their concepts of editor and the likes via mediate inference and practical learning over time on different graphic software of their acquaintance, will find themself discomforted and confused working with Blender. Arguably, the amount of overlap of unicepts between people is a matter of degrees, in the case of software we may have groups of users where the amount of sharing is high, and other gropus where is it minimal. We believe that the case of Blender is rather unexceptional, that in most complex software there can be examples of unicepts, that may or may not be succesfully preserved in their integrity, but can be alien to unicepts named with the same words in other software.

4. Conclusions

This article takes on the issue of how conceptual integrity in software design can be pursued, under the assumption that there is no such thing as "concepts". We addressed this issue not as a logic exercise of counterfactual reasoning, but because recently a few distinguished philosophers have argued against the very existence of concepts. In trying to articulate the consequences for conceptual integrity, we adopted one of the best alternatives, in our view, proposed so far: Millikan's "unicept". It is a mental entity that is built by collecting and integrating information about some thing of the world, in a large variety of ways, and differs substantially from concepts in that unicepts are not shared as such by people. We analyzed the case of Blender software, and the items ordinary called "concepts" that characterize its user interface, arguing that they are better construed as "unicepts", because, despite the same verbal labels, differ intrinsically from those used in other graphics software. However, the lack of generality does not hamper the software design to pursue a high degree of integrity, that now we can call *uniceptual integrity*. It is probably an integrity less ambitious of that fostered by Frederick Brooks, that works well for group of users that share, at least, an important part of the unicepts founding the design, and not for others. It is our opinion that the reasoning carried out in this paper has a relevance for application in software engineering, especially when large teams of developer are involved and the conceptual integrity issues have a significant impact.

References

[1] L. W. Barsalou. Perceptual symbol systems. *Behavioral and Brain Science*, 22:577–660, 1999.

[2] G. A. Blaauw and F. P. Brooks. *Computer Architecture: Concepts and Evolution*. Addison Wesley, Reading (MA), 1997.

[3] A. Brito. *Blender 3D: Architecture, Buildings, and Scenery: Create photorealistic 3D architectural visualizations of buildings, interiors, and environmental scenery*. Packt Publishing Ltd, Birmingham (UK), 2008.

[4] F. Brooks. *The Mythical Man-Month: Essays on Software Engineering*. Addison Wesley, Reading (MA), 1975.

[5] L. Brooks. Nonanalytic concept formation and memory for instances. In E. Rosch and B. B. Lloyd, editors, *Cognition and concepts*, pages 169–211. Lawrence Erlbaum Associates, Mahwah (NJ), 1978.

[6] S. Carey. *Conceptual change in childhood*. MIT Press, Cambridge (MA), 1985.

[7] C. Carpineto and G. Romano. Using concept lattices for text retrieval and mining. In *Formal Concept Analysis – Foundations and Applications* [11], pages 161–179.

[8] F. Détienne. *Software design: cognitive aspects*. Springer-Verlag, Berlin, 2002.

[9] J. Dvorak. Conceptual entropy and its effect on class hierarchies. *IEEE Computer*, 27:59–63, 1994.

[10] J. Fodor. Concepts: A potboiler. *Cognition*, 50:95–113, 1994.

[11] B. Ganter, G. Stumme, and R. Wille. *Formal Concept Analysis – Foundations and Applications*. Springer-Verlag, Berlin, 2005.

[12] B. Ganter and R. Wille. *Formal concept analysis: mathematical foundations*. Springer-Verlag, Berlin, 1999.

[13] R. L. Goldstone and A. Kersten. Concepts and categorization. In I. B. Weiner, editor, *Handbook of psychology*. John Wiley, New York, 2003.

[14] J. Hampton. Prototype models of concept representation. In I. Van Mechelen, J. Hampton, R. S. Michalski, and P. Theuns, editors, *Categories and concepts: Theoretical views and inductive data analysis*, pages 67–95. Academic Press, New York, 1993.

[15] D. Hume. *An Enquiry Concerning Human Understanding*. A. Millar, London, 1748.

[16] D. Jackson. Towards a theory of conceptual design for software. In *ACM International Symposium on New Ideas, New Paradigms and reflections on programming and software – Onward!*, pages 282–296. ACM, 2015.

[17] J. Katz and J. Fodor. The structure of a semantic theory. *Language*, 39:170–210, 1963.

[18] L. K.Komatsu. Recent views of conceptual structure. *Psychological Bulletin*, 112:500–526, 1992.

[19] J. Locke. *An essay concerning human understanding*. Meridian Books, Cleveland, 1690.

[20] E. Machery. Concepts are not a natural kind. *Philosophy of Science*, 72:444–467, 2005.

[21] E. Machery. *Doing without concepts*. Oxford University Press, Oxford (UK), 2009.

[22] E. Margolis and S. Laurence, editors. *Concepts: Core readings*. MIT Press, Cambridge (MA), 1999.

[23] E. M. Markman. *Knowledge representation*. Lawrence Erlbaum Associates, Mahwah (NJ), 1999.

[24] R. G. Millikan. *On Clar and Confused Ideas: An Essay About Substance Concepts*. Cambridge University Press, Cambridge (UK), 2000.

[25] R. G. Millikan. An epistemology for phenomenology? In R. Brown, editor, *Consciousness Inside and Out: Phenomenology, Neuroscience, and the Nature of Experience*, pages 13–26. Springer-Verlag, Berlin, 2014.

[26] G. L. Murphy. *The big book of concepts*. Cambridge University Press, Cambridge (UK), 2002.

[27] C. Peacocke. *A study of concepts*. MIT Press, Cambridge (MA), 1992.

[28] G. Piccinini and S. Scott. Splitting concepts. *Philosophy of Science*, 75:390–409, 2006.

[29] J. Prinz. *Furnishing the Mind – Concepts and Their Perceptual Basis*. MIT Press, Cambridge (MA), 2002.

[30] L. J. Rips. The current status of research on concept combination. *Minds and Language*, 10:72–104, 1995.

[31] E. Rosch. Principles of categorization. In E. Rosch and B. Lloyd, editors, *Cognition and Categorization*. Lawrence Erlbaum Associates, Mahwah (NJ), 1978.

[32] K. L. Smoke. An objective study of concept formation. *Psychological Monographs*, 42:1–46, 1932.

[33] K. Solomon, D. L. Medin, and E. Lynch. Concepts do more than categorize. *Trends in Cognitive Sciences*, 3:99–105, 1999.

[34] T. Tilley, R. Cole, P. Becker, and P. Eklund. A survey of formal concept analysis support for software engineering activities. In *Formal Concept Analysis – Foundations and Applications* [11], pages 250–271.

EXTRACTION OF PATTERNS USING NLP: GENETIC DEAFNESS[1]

Anabel Fraga[1], Javier Garcia[1], Eugenio Parra[1], Valentín Moreno[1]

[1] *Computer Science Department, Carlos III of Madrid University*
Av. Universidad 30, Leganés, Madrid, Spain
{afraga, eugenio.parra@kr.inf.uc3m.es, vmoreno}@inf.uc3m.es

Keywords: indexing, ontologies, knowledge, genetic deafness, medicine, reuse, retrieval, conceptual text integrity.

Abstract: In the domain of Genetic Deafness in medicine, it is important to detect some patterns. Medical doctors needs to search and deal with information from diverse sources and it is important to be able to cross information between sources. As part of a solution, one approach to minimize the impact of this lack and increase the success of the retrieval process crossing diverse sources of information laid in the use of Natural Language Processing techniques permitting conceptual integrity of text.

1 INTRODUCTION

In this research project, we solved the following problem: is it possible to make a computer recognize linguistic patterns in a document? For example, when reading the sentence:
"The patient had a temperature of 102F at Saturday"
With syntactic tokens: "Determiner + Noun + Verb + Preposition + Noun + Preposition + Noun + Preposition + Noun"
Is it possible for the computer to understand it both syntactically and semantically? Moreover, once this information is acquired, is it possible to generate patterns that allow recognizing similar sentences throughout a document?

With this problem as a starting point, this project is related to the studies of Natural Language Processing (NLP). For that, by using vocabulary from a specific domain, the computer will read a number of documents and automatically generate all linguistic patterns contained in them. Therefore, there are two main objectives:

- Acquire the terminology from a specific domain.

[1] DOI: 10.18293/SEKE2017-204

• Perform the linguistic pattern learning process by reading and analyzing a number of documents from the same domain.

The chosen domain for this project was "Genetic Deafness" (medicine), and the language used was English.

This paper is a summary of the project, which covers the followed work procedure and the results of the experimentation, as well as the conclusions reached after the process was completed.

There were two potential motivations for this research project:

• Effective information extraction from a text document. When a human reads a sentence, he or she extracts a piece of the information that he or she considers useful. See the previous example: "The patient had a temperature of 102F at Saturday". From this sentence, we can retain the information related to the temperature (102F) and/or the time when the measure was taken (at Saturday).

For a computer, this "understanding" process is difficult. There are no trivial algorithms that, after reading the sentence, can storage the knowledge received about the patient's temperature and the day the measure was taken. A recurrent solution for this problem is to use templates and file formats, which allows the computer to receive all the information without actually understanding it. This, however, forces the information to be pre-processed so it can meet software specifications, which consumes more time and effort.

A previous learning of linguistic patterns may solve this problem. If the computer is able to recognize the example sentence above as a learned pattern, then it would realize that the seventh word indicates the temperature (102F) and the last one indicates the day (Saturday).

Therefore, the pattern analysis could be useful to acquire all relevant information from a document of variable length.

• Writing style guide development. In several fields, like law or medicine, there are specific writing styles to follow. Therefore, the results from this project in a specific domain can be useful in a didactic way. For example: which word families are more frequent in the domain? Which linguistic sequences are more common? Is there a recognizable conceptual integrity of the text in a higher-level than individual concepts and even recurring patterns? And so on.

By studying the linguistic patterns from a domain, we can contribute to the learning of the writing style expected in a document, in order to show all the information in an organized state.

This is also helpful for computers: when the writing style is standardized, the pattern recognition is easier and, therefore, the information retrieval is easier and more efficient.

2 STATE OF THE ART AND RELATED WORK

2.1 How to Define Conceptual Integrity of Texts?

Conceptual Integrity has been defined by Brooks [ref], mainly in the context of software systems design and development. In this paper we deal with texts, which might be software assets, such as software documentation in several domains, but are certainly different than software system design tools or runnable software itself.

Since our main research goal of this paper is to enable mining of relevant texts for a given domain – in our case study it is "Genetic Deafness" – one asks: what is the relevant definition of "Conceptual Integrity" in this context?

A preliminary proposal for such a definition would include the following steps:

• Choose a basic set of concepts to characterize the desired texts in the desired domain;

• Select a set of patterns – to be obtained and explained later on in the research of this paper – containing the basic set of concepts;

• Determine reasonable minimal thresholds, to check whether the appearance of the chosen concepts and patterns are above these reasonable thresholds.

In our investigation, as future work, we shall make extensive tests of these steps for certain chosen domains.

2.2 Information Reuse

Reuse in software engineering is present throughout the project life cycle, from the conceptual level to the definition and coding requirements. This concept is needed to improve the quality and optimization of the project development, but it has difficulties in standardization of components and combination of features. Also, the software engineering discipline is constantly changing and updating, which quickly turns obsolete the reusable components (Llorens, 1996).

The patterns are fundamental reuse components that identify common characteristics between elements of a domain and can be incorporated into models or

defined structures that can represent the knowledge in a better way.

2.3 Natural Language Processing

The need for implementing Natural Language Processing techniques (see Figure 1) arises in the field of the human-machine interaction through many cases such as text mining, information extraction, language recognition, language translation, and text generation, fields that requires a lexical, syntactic and semantic analysis to be recognized by a computer (Cowie et al., 2000). The natural language processing consists of several stages which take into account the different techniques of analysis and classification supported by the current computer systems (Dale, 2000).

1) **Tokenization**: The tokenization corresponds to a previous step on the analysis of the natural language processing, and its objective is to demarcate words by their sequences of characters grouped by their dependencies, using separators such as spaces and punctuation (Moreno, 2009).

2) **Lexical Analysis**: Lexical analysis aims to obtain standard tags for each word or token through a study that identifies the turning of vocabulary, such as gender, number and verbal irregularities of the candidate words (Hopcroft et al., 1979).

3) **Syntactic Analysis**: The goal of syntactic analysis is to explain the syntactic relations of texts to help a subsequent semantic interpretation (Martí et al., 2002), and thus using the relationships between terms in a proper context for an adequate normalization and standardization of terms.

4) **Grammatical Tagging**: Tagging is the process of assigning grammatical categories to terms of a text or corpus. Tags are defined into a dictionary of standard terms linked to grammatical categories (nouns, verbs, adverb, etc.), so it is important to normalize the terms before the tagging to avoid the use of non-standard terms (Weischedel et al., 2006). Grammatical tagging is a key factor in the identification and generation of semantic index patterns, in where the patterns consist of categories not the terms themselves. The accuracy of this technique through the texts depends on the completeness and richness of the dictionary of grammatical tags.

5) **Semantic and Pragmatic Analysis**: Semantic analysis aims to interpret the meaning of expressions, after on the results of the lexical and syntactic analysis. One of the main challenges NLP has tried to overcome since its beginnings is the problem of ambiguity.

3 METHODOLOGY

The methodology of this project is based on the usage of concepts related to genetic deafness in order to recognize the most frequent patterns from a number of documents while facing the problems and limitations from the Natural Language Processing.

In this project, the BoilerPlates tool has been used. This tool, developed by the Carlos III de Madrid research team (Knowledge Reuse Research Team), can perform the analysis of linguistic patterns given a previously learned set of concepts and grammar categories. Its database already contains vocabulary that is not related to any domain (such as the verb “to be”, prepositions, adverbs…). However, to effectively use this tool in a certain domain, the database must be updated with domain-specific concepts prior to the pattern extraction.

The methodology used for this project is divided in the following steps:

1. Acquisition and processing of domain documents.
2. Extraction of related terminology.
3. Adaptation of terminology to BoilerPlates.
4. Using the BoilerPlates tool. It has been used. This tool, developed by the Carlos III de Madrid research team (Knowledge Reuse Research Team), can perform the analysis of linguistic patterns given a previously learned set of concepts and grammar categories. Its database already contains vocabulary that is not related to any domain (such as the verb “to be”, prepositions, adverbs…). However, to effectively use this tool in a certain domain, the database must be updated with domain-specific concepts prior to the pattern extraction.
5. Extraction of results.
6. Exposition and analysis of results.

4 RESULTS

Eight scenarios were identified for this study, with minimal frequencies from 1 to 20without or with semantic distinction. Once the scenarios were individually analysed, the combined results were as it follows:

Number of patterns identified

As expected, the lower the minimal frequency (MF) value chosen, the more patterns are considered in the identification process.

The growth of the number of patterns found is inversely exponential: in the experiments with MF established to 1, about 140.000 patterns were found.

When raising the MF to 5, the number drops at 14.500 on average, which is about a 90% decrease on patterns found. On the experiments with MF set to 10 and 20, the number of patterns were approximately 5.600 and 2.800 respectively, which is a considerable drop but less severe than before.

Patterns related to genetic deafness

After analysing the results as a whole set of patterns, a subset was created for each scenario, containing only those patterns which has at least one concept related to genetic deafness ("domain patterns").

This subset was a minority in every studied case. For all scenarios, about 10 to 16% of the patterns contained at least one concept from the domain. This was predictable, as most of the words contained in one document about a certain topic are not terminology from the domain, such as prepositions, numbers or unrelated nouns. It is important for conceptual integrity.

Conclusions

As Manning said and we are investigating in our hypothesis, it is possible to get structure and regularity in a set of documents in order to achieve a regular pattern to write or suggest better manners to prepare documents with a higher complexity. Natural language is complex and its regularity depends on the domain we are dealing with. The more regular and mature the domain, the best it is to extract a set of patterns. The more structure the more effective the set of patterns.

Once the analysis of the results was completed, we reached some of the following conclusions regarding the pattern identification and studio in the domain of genetic deafness:

1. The minimal frequency value made a considerable impact in the analysis. The higher the value, the lower the number of identified patterns, since a sequence must reach a bigger number of repetitions until it is considered a pattern.
2. On the matter of semantic distinction: The first experiments showed that there was little difference whether applying semantics or not. However, when the weighted study was completed, it was discovered that semantics played a considerable role during the analysis, such as which patterns were considered the most valuable.
3. Of all patterns discovered, around 10% to 16% had at least one domain concept.
5. The most frequent concepts that are related to "genetics" were those that are about gene manipulation and mutation.

References

COWIE, Jim. WILKS, Yorick. Information Extraction. En DALE, R. (ed). Handbook of Natural Language Processing. New York: Marcel Dekker, 2000. Pp.241-260.

DALE, R. Symbolic Approaches to Natural Language Processing. En DALE, R (ed). Handbook of Natural Language Processing. New York: Marcel Dekker, 2000.

FRAGA, Anabel. A methodology for reusing any kind of knowledge: Universal Knowledge Reuse. PhD Disertation. Universidad Carlos III de Madrid, 2010.

FRAGA et al. SYNTACTIC-SEMANTIC EXTRACTION OF PATTERNS APPLIED TO THE US AND EUROPEAN PATENTS DOMAIN. SKY2016 / IC3K2016. Portugal, 2016.

HOPCROFT, J.E. ULLMAN, J.D. Introduction to automata theory, languages and computations. Addison-Wesley, Reading, MA, United States. 1979.

LLORENS, J., Morato, J., Genova, G. RSHP: An Information Representation Model Based on Relationships. In Ernesto Damiani, Lakhmi C. Jain, Mauro Madravio (Eds.), Soft Computing in Software Engineering (Studies in Fuzziness and Soft Computing Series, Vol. 159), Springer 2004, pp. 221-253.

LLORENS, Juan. Definición de una Metodología y una Estructura de Repositorio orientadas a la Reutilización: el Tesauro de Software. Universidad Carlos III. 1996.

MANNING Christopher, "Foundations of Statistic Natural Language Processing ", Cambridge University, England, 1999, 81

MARTÍ, M. A. LLISTERRI, J. Tratamiento del lenguaje natural. Barcelona: Universitat de Barcelona, 2002. p. 207.

MARTIN, James N. 1996. Systems Engineering Guidebook: A Process for Developing Systems and Products. CRC Press, Inc.: Boca Raton.

MORENO, Valentín, Pablo Miguel Suárez, Anabel Fraga, Juan Llorens, and Eugenio Parra. 2013. Método de generación de patrones semánticos. PCT/ES2013/070638, issued 2013.

MORENO, Valentín. Representación del conocimiento de proyectos de software mediante técnicas automatizadas. Anteproyecto de Tesis Doctoral. Universidad Carlos III de Madrid. Marzo 2009.

PARRA, Eugenio. 2016. Metodología orientada a la optimización automática de la calidad de los requisitos. PhD

PARRA, Eugenio. Metodología orientada a la optimización automática de la calidad de los requisitos. PhD Disertation. Universidad Carlos III de Madrid, 2016.

WEISCHEDEL, R. METTER, M. SCHWARTZ, R. RAMSHAW, L. PALMUCCI, J. Coping with ambiguity and unknown through probabilistic models. Computational Linguistics, vol. 19, pp. 359-382.

Extending Software Systems While Keeping Conceptual Integrity

Reuven Yagel
Software Engineering Department
Azrieli – Jerusalem College of Engineering
Israel
robi@jce.ac.il

Abstract— **Design and analysis of software systems in terms of their Conceptual Integrity is a demanding task. Nonetheless, progress has been made in recent years and actual software systems in practical use, such as Git, have been analyzed. In this work we make a further first step within the conceptual analysis approach, by asking how to extend software systems by addition of further components while keeping Conceptual Integrity of the resulting system. We propose specific techniques to this end. As a case study to illustrate these techniques, we analyze a popular project management service, namely Gitlab, for its various services integrity and adaptability to software engineering lifecycle stages.**

Keywords- Component; Software Repository; Lifecycle Management; Conceptual Integrity; Git; Gitlab

I. INTRODUCTION

Conceptual Integrity has been identified by Brooks [1, 2] as the main challenge for software system design. More recently, Jackson et. al. [3, 4] further formulated and demonstrated the Conceptual design and analysis approach for a popular software system – git [5]. In this paper we build on this work and start broadening this view by examining the further issue of extending software systems with additional components while preserving the Conceptual Integrity of the whole system. We propose some specific techniques in this respect and illustrate them by analyzing popular code repository and project management services.

A. Git Based Software Project Management Systems

The software development industry is going through major changes in recent years. Among them are new tools and services for software project management (SPM) such as github.com [6]. This service, taken as an example, is based on cloud hosting of git – a distributed version control system. Beyond managing source code versions, the service simplifies workflows of branching and merging and also includes various project management tools, in particular social features, which allow vast collaboration options. A variety of software organizations and their offered services, have different service models. Examples of the referred kind are SourceForge, Microsoft's Team Foundation Services, bitbucket, Coding.net and many others.

In this work we focus on such a similar service – gitlab.com [8]. It is also a software project management service centered on git. Beyond managing source code versions, it includes various project management tools for communication, documentation, testing and more.

The reason for choosing git-centered software systems to illustrate our proposal, is to capitalize on existing analyses of Git and Gitless by Jackson and co-workers [3], [4], and their wide industry usage and acceptance.

B. Paper Organization

The remainder of this paper is organized as follows. In section II we review related literature. In section III we describe a general Lifecycle Model underlying the software systems used to illustrate our approach. In section IV, specific techniques of our approach to extend software systems while keeping conceptual integrity are proposed. In section V the Gitlab software system serves to illustrate our proposed approach. The paper is concluded with a short discussion in section VI.

II. RELATED LITERATURE

Brooks [1, 2] suggested three principles for representing the notion of conceptual integrity:

- Orthogonality
- Propriety
- Generality

Jackson et. al. [3] showed how git conceptual design is quite complex and also problematic in light of these principles. They go further on and suggest a new design, titled gitless to correct those issues.

Git [5] is an open source distributed version control system. It became quite popular in recent years, but still criticized for its usability and conceptual integrity issues.

Models of software development as well as processes, methods, and tools are widely discussed. Here we reference mainly Rajlich [8], but many others are discussing those, e.g., [11-15].

III. LIFECYCLE MODEL

Since the discussed services are all aimed for helping developing large software projects, we choose here a general

DOI: 10.18293/SEKE2017-205

model for software project lifecycle adapted from Rajlich [8]. It is a basic and general staged model of software "lifespan", which includes the stages of:

1. *Initial development,*
2. *Evolution,*
3. *Maintenance* (servicing),
4. *Phase-out,*
5. *Close-down.*

The second and third stages are usually iterative and incremental (see Figure 1 of [8]). A Software Project Management System is expected to support development based on such a model and its various derivatives or alternatives.

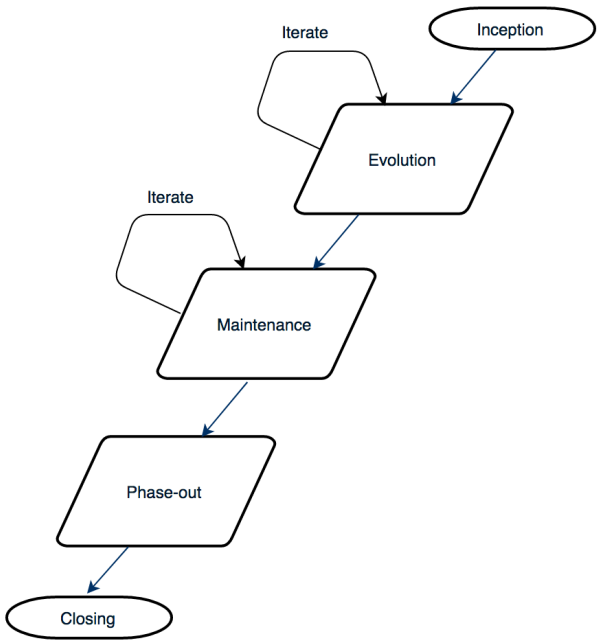


Figure 1 Staged model of software adapted from [8]

IV. Our Approach: Software Extension with Conceptual Integrity Preservation

Our approach consists of enabling addition of services along the lifecycle of a software system, while having an infrastructure to preserve Conceptual Integrity.

Typically, most SPMs contain at least the following services:

1) Version Control for code/software (SCM)
2) Documentation system
3) Issue Database
4) API and integration points with other services, e.g., continuous integration (CI).

We now formulate our expectations from a SPM system in the light of the above mentioned conceptual integrity concepts. In other words, can we forecast eventual Conceptual Integrity violations from these typical services?

- *Orthogonality* – in principle each of the above services should be used standalone, although in practice users will expect integration between them. As an example: there might be an open issue to fix some missing documentation. Upon update of the documentation (2), the version control system is preserving a commit (1), the issue changes status to closed (3) and a CI service is triggered to run tests (4).

Here is the analysis:

- *Propriety* – the set of services above is quite concise and centered around project management. Extensions are mainly through 3rd party software integrations.
- *Generality* – each service can be adapted to a specific project lifecycle and the actual project being carried. For example, version control is not limited to a specific programming language (or even software artifacts at all). The use of git, opens the door to many possible git branching models [9].

From this preliminary analysis, the requirements for a Conceptual Integrity preservation infrastructure are formulated as specific design patterns for Conceptual Integrity to be prepared in advance:

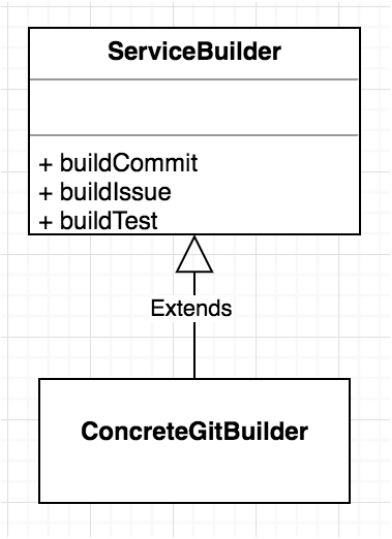


Figure 2: Using Builder Concatenation Pattern

Concatenation patterns – design patterns that can be instantiated for concatenation of services, as in the above example, which illustrates the Orthogonality principle. These patterns could use

for example the "builder" GoF [10] pattern (shown in Fig. 2), suitable for setting up a related set of services.

Integration interfaces – this set of patterns should act as a sort of "adapter" or "façade" GoF patterns, with the purpose of integrating yet unknown 3rd party software.

Models and languages variations – these patterns needed to cope with variation of programming languages and say branching models, should resemble GoF "strategy" patterns.

V. The Gitlab Case Study

We shortly discuss the Gitlab services, and their conformance to conceptual integrity principles. The Gitlab feature page [7] presents the software as containing "Powerful features for modern software development, tightly integrated into one platform". It is centered around a hosted git server with a rich web interface.

- *Version Control* – Gitlab suggests hosting of unlimited private or public git repositories, thus conforming to generality. The web platform is used to provide accessible and convenient graphical user interfaces for the various version control operations. Gitlab can be installed on premise or consumed as a cloud service. Still the usability issues of git are not totally masked, and this might explain the slow adoption rate of such a service outside of the software engineering community.
  Gitlab has a snippet service for sharing portions of code. Such a service could be attitude as an extension to version control; alas a snippet is not versioned!
  Gitlab added fined grain access control (FGAC) and several privacy tracks, which in fact are evidently options for version control especially for enterprise settings. Specific files or folders can be locked in order to prevent merge conflicts. Gitlab also suggest sharing small portions of code by a code snippet service.
- *Documentation* – Gitlab provides a wiki system and also a static site hosting solution (Gitlab pages). The wiki system is maintained in a separate and less accessible git repository instance. This is, in our mind, a conceptual integrity issue, since project artifacts are handled differently (github has the same issue). Also, it is a common practice to have a major documentation file – usually named *Readme*, which resides in the main source tree. Thus, project documentation become spread in at least three different areas – source code, wiki and a static site, not to mention other possible documentations (formal or informal).
- *Issue Database* – the text of an issue can link to other artifacts, and there are ways to change issue statuses from version control commit messages. This is another potential point were conceptual integrity can be weakened. Gitlab also has an activity stream, so users need to adjust their most efficient project update communication patterns.
- *API and integrations with other services, e.g., continuous integration (CI)* – some of the services are hosted and some are just interfaces to 3rd parties. An infrastructure can help define the interfaces in more concise ways to improve conceptual integrity.

VI. Discussion and Future Work

In this position paper, we outlined an ongoing research to examine the conceptual integrity of complex software systems and laid directions for patterns for conforming to integrity principles. We intend to further analyze in detail Gitlab and compare it to other services. The expected outcome of the analysis and comparisons with other services is a suggested series of concrete detailed improvements of the system.

The economics of the competition between companies suggesting those services might lead to "feature creep" followed by breaking conceptual integrity. This is indirectly demonstrated in the long feature lists in the providers' websites.

Conceptual integrity will also be discussed from other angles, e.g., operations, hosting options, and pricing models.

Acknowledgment: I would like to thank Iaakov Exman for helpful and fruitful advice.

References

[1] F.P. Brooks, *The Mythical Man-Month – Essays in Software Engineering* – Anniversary Edition, Addison-Wesley, Boston, MA, USA, 1995.

[2] F.P. Brooks, *The Design of Design: Essays from a Computer Scientist*, Addison-Wesley, Boston, MA, USA, 2010.

[3] D. Jackson and S. P. DeRosso, "What's wrong with git?: a conceptual design analysis" in Proceedings of the ACM international symposium on New ideas, new paradigms, and reflections on programming & software (Onward!), 2013.

[4] 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.

[5] S. Chacon, Pro Git, Apress 2009, http://git-scm.com/book

[6] Github, https://github.com/about

[7] Gitlab, https://about.gitlab.com/features/

[8] V. Rajlich, "Software Evolution and Maintenance", in Proceedings of the on Future of Software Engineering (FOSE), 2014.

[9] Atlassian Tutorial, "Comparing Workflows", https://www.atlassian.com/git/tutorials/comparing-workflows

[10] E. Gamma et. al., Design Patterns: Elements of Reusable Object-Oriented Software, AddisonWesley Professional, 1994.

[11] Amber, The Agile System Development Life Cycle (SDLC), http://www.ambysoft.com/essays/agileLifecycle.html

[12] P. A. Laplante, What Every Engineer Should Know about Software Engineering, CRC Press, 2007.

[13] R. S. Pressman, Software Engineering: A Practitioner's Approach, 8th Edition, McGraw-Hill Education, 2014.

[14] B. Boehm, "A Spiral Model of Software Development and Enhancement". In: ACM SIGSOFT Software Engineering Notes (ACM) 11(4):14-24, 1986.

[15] S. McConnell, "7: Lifecycle Planning", in Rapid Development: Taming Wild Software Schedules. Microsoft Press., 1996.

A Comparative Study of Software Bugs in Clone and Non-Clone Code

Judith F. Islam Manishankar Mondal Chanchal K. Roy Kevin A. Schneider
Department of Computer Science, University of Saskatchewan, Canada
{judith.islam, mshankar.mondal, chanchal.roy, kevin.schneider}@usask.ca

***Abstract*—Code cloning is a recurrent operation in everyday software development. Whether it is a good or bad practice is an ongoing debate among researchers and developers for the last few decades. In this paper, we conduct a comparative study on bug-proneness in clone code and non-clone code by analyzing commit logs. According to our inspection on thousands of revisions of seven diverse subject systems, the percentage of changed files due to bug-fix commits is significantly higher in clone code compared with non-clone code. We perform a Mann-Whitney-Wilcoxon (MWW) test to show the statistical significance of our findings. Finally, the possibility of severe bugs occurring is higher in clone code than in non-clone code. Bug-fixing changes affecting clone code should be considered more carefully. According to our findings, clone code appears to be more bug-prone than non-clone code.**

I. Introduction

If two or more code fragments in a software system's code-base are exactly or nearly similar to one another we call them code clones [1], [2]. A group of similar code fragments forms a clone class. Code clones are mainly created because of the frequent copy/paste activities of programmers during software development and maintenance [1] [1]

A significant number of studies [3]–[21] have been conducted on discovering the impact of cloning on software maintenance. While a number of studies [3], [6], [7], [9]–[12] have revealed some positive sides of code cloning, there is strong empirical evidence [4], [5], [8], [13]–[15], [17], [19]–[21] of negative impacts of code clones too. These negative impacts include higher instability [15], late propagation [4], and unintentional inconsistencies [5]. Existing studies [4], [22] show that code clones are related to bugs in the code-base.

Several studies have showed that bugs have a great effect on code in software systems. Our previous study [23] provides evidence of bug-replication in clone code. Also, Sajnani et al. [24] showed that cloned code has less problematic bug patterns than non-cloned code. They have used bugs reported by FindBugs from just one snapshot of the last revision of the system. Whereas we consider bugs reported during the evolution of a software system through thousands of commits. In that paper [24], they worked on tool reported bugs whereas we work on the developer reported bugs. Moreover, they have considered only Java programming language whereas we work on two programming languages, C and Java. These issues motivate us to work on bug reports generated by developers to see the impact of bug-fix commits on both clone code and non-clone code. We consider bug-fixing commits reported by the developers from thousands of commits in open source projects.

To explore the effects of bug-fix changes between clone and non-clone code, we conduct a comparative study. We consider thousands of revisions of seven diverse subject systems written in two different programming languages (Java and C). We detect code clones from each of the revisions of a subject system using the NiCad [25] clone detector, analyze the evolution history of these code clones, and investigate whether and to what extent they contain bugs. To find non-clone bug-fix commits we first identify all the commits that are related to fixing a bug. Among these bug-fix commits we detect those which have clone code. We consider the remaining bug-fix commits as non-clone bug-fix commits. We automatically count the total number of files that contain changes in source code. Among these files we detect those files which have changes in clone code. Omitting these files from the total files we get the changes in non-clone code. Then we calculate percentages of changes for both clone and non-clone code. We found that the percentage of changed files containing clone code is significantly higher than non-clone code. We validate our findings using the Mann-Whitney-Wilcoxon (MWW) test for three types of clones with non-clone code.

We investigate the three research questions listed in Table I. We find that the percentage of files changed due to bug-fix commits is significantly higher in clone code compared with non-clone code. Moreover, the percentage of files that have changes in Type 1 and Type 2 clone code is higher than changes in Type 3 clone code. This findings can be used for ranking of clone code. Also, the occurrence of severe bugs is more in clone code than non-clone code.

The rest of the paper is organized as follows. Section II contains the terminology, Section III discusses the experimental steps, Section IV answers the research questions by presenting and analyzing the experimental results, Section V discusses the related work, Section VI discusses possible threats to validity, and Section VII concludes the paper and discusses possible future work.

II. Terminology

We conduct our analysis considering both exact (Type 1) and near-miss clones (Type 2 and Type 3 clones) [1], [2]. The clone-types are defined below.

Type 1 Clones. If two or more code fragments in a particular code-base are exactly the same disregarding the

DOI reference number: 10.18293/SEKE2017-056

TABLE I
RESEARCH QUESTIONS

SL	Research Question
RQ 1	What percentage of files get affected because of clone and non-clone bug-fix commits?
RQ 2	How often do bug-fix changes occur to the clone and non-clone code?
RQ 3	Is there any difference between the severity of the bugs occurring in clone and non-clone code?

TABLE II
SUBJECT SYSTEMS

Systems	Lang.	Domains	LLR	Revisions
Ctags	C	Code Def. Generator	33,270	774
Camellia	C	Image Processing Library	89,063	170
Brlcad	C	Solid Modeling CAD	39,309	735
jEdit	Java	Text Editor	191,804	4000
Freecol	Java	Game	91,626	1950
Carol	Java	Game	25,091	1700
Jabref	Java	Reference Management	45,515	1545

LLR = LOC in the Last Revision

comments and indentations, these code fragments are called exact clones or Type 1 clones of one another.

Type 2 Clones. Type 2 clones are syntactically similar code fragments in a code-base. In general, Type 2 clones are created from Type 1 clones because of renaming of identifiers and/or changing of data types.

Type 3 Clones. Type 3 clones are mainly created because of additions, deletions, or modifications of lines in Type 1 or Type 2 clones. Type 3 clones are also known as gapped clones.

III. EXPERIMENTAL STEPS

We conduct our research on seven subject systems (three C and four Java systems). We consider these seven subject systems since these systems have variations in application domains, sizes, revisions and also used by our other studies. These subject systems are listed in Table II which were downloaded from the SourceForge online SVN repository [26]. In this table, the total number of revisions of each subject system is given along with the lines of code (LOC) in the last revision.

A. Preliminary Steps

We perform the following steps for detecting fixed bugs: **(1)** Extraction of all revisions (as stated in Table II) of each of the subject systems from the online SVN repository; **(2)** Detection and extraction of code clones from each revision by applying NiCad [25] clone detector; **(3)** Detection of changes between every two consecutive revisions using diff; **(4)** Locating these changes to the already detected clones of the corresponding revisions; and **(5)** Detection of bug-fix commit operations. For completing the first four steps we use the tool SPCP-Miner [27]. We will describe the detection of bug-fix commits later in this section. In Section IV we will describe how we detect bug-fix changes in clone and non-clone code.

We use NiCad [25] for detecting clones since it can detect all major types (Type 1, Type 2, and Type 3) of clones with high precision and recall [28], [29]. Using NiCad we detect block clones including both exact (Type 1) and near-miss (Type 2, Type 3) clones of a minimum size of 10 LOC with 20% dissimilarity threshold and blind renaming of identifiers. NiCad settings for detecting three clone-types (Type 1, Type 2, and Type 3) are shown in Table III. For different settings of a clone detector the clone detection results can be different and thus, the findings on bugs in code clones can also be different. Hence, selection of appropriate settings (i.e., detection parameters) is important. We used the mentioned settings in our research, because [30] Svajlenko and Roy show that these settings provide us with better clone detection results in terms of both precision and recall. Moreover, code clones with a minimum size of 10 LOC are more appropriate from maintenance perspectives [1], [31], [32]. Before using the NiCad outputs of Type 2 and Type 3 cases, we processed them in the following way.

TABLE III
NICAD SETTINGS FOR THREE TYPES OF CLONES

Clone Types	Identifier Renaming	Dissimilarity Threshold
Type 1	none	0%
Type 2	blindrename	0%
Type 3	blindrename	20%

(1) Every Type 2 clone class that exactly matched any Type 1 clone class was excluded from Type 2 outputs.

(2) Every Type 3 clone class that exactly matched any Type 1 or Type 2 class was excluded from Type 3 outputs.

We processed NiCad clone detection results in the mentioned ways because we wanted to investigate bug in three types of clones separately.

B. Bug-proneness Detection Technique

For each subject system, we first retrieve the commit messages by applying the 'SVN log' command. A commit message describes the purpose of the corresponding commit operation. We automatically infer the commit messages using the heuristic proposed by Mockus and Votta [33] in order to identify those commits that occurred for the purpose of fixing bugs. Then we identify which of these bug-fix commits make changes to clone fragments. If one or more clone fragments are modified in a particular bug-fix commit, then it is an implication that the modification of those clone fragment(s) was necessary for fixing the corresponding bug. In other words, the clone fragment(s) are related to the bug. In this way we examine the commit operations of a candidate system, analyze the commit messages to retrieve the bug-fix commits, and identify those clone fragments that are related to the bug-fix. We also determine the number of changes that occurred to such a clone fragment in a bug-fix commit using the UNIX *diff* command.

The procedure that we follow to detect the bug-fix commits was also previously followed by Barbour et al. [4]. Barbour et al. [4] detected bug-fix commits in order to investigate whether late propagation in clones is related to bugs. They at

first identified the occurrences of late propagations and then analyzed whether the clone fragments that experienced late propagations are related to a bug-fix. In our study we detect bug-fix commits in the same way as they detected, however, our study is different in the sense that we investigate the bugs of different types of code clones. Also, Barbour et al. [4] did not investigate the most important clone type, the Type 3. Generally, the number of Type 3 clones in a system is the highest among the three clone-types. We consider Type 3 clones in our bug-fix study.

IV. Experimental Results and Analysis

We mention our three research questions in Table I. In this section we present our experimental results and analyze them to find the answers to our research questions.

A. Answering the first research question (RQ 1)

RQ 1: *What percentage of files get affected because of clone and non-clone bug-fix commits?*

Motivation. It is important to know the percentage of affected files due to bug-fixing commits and compare between clone and non-clone code. More affected files means more changes in the system. More attention is needed when more changes occur. Knowing the information we can emphasize on which type of code (clone or non-clone) are affecting the system more.

Methodology. To answer this research question we automatically count the total number of files that contain clone code in three different types of clones (Type 1, Type 2 and Type 3) and total number of files containing non-clone code. Also, we detect the total number of files that contain changed clone in three different clone types and the total number of files containing changed non-clone code. Then we calculate their percentages for individual clone types.

FC: This is the total number of files that have clone code. Columns with the heading FC in Table IV represents the value.

FCC: This is the total number of files that contain changed clone code. This value is given in columns with the heading FCC in Table IV.

PFCC: To find out the percentage of the number of files that have changed clone code we use following equation for all subject systems. In Table IV columns with the heading PFCC show this value. Equation 1 shows the assessment of the percentages.

$$PFCC = \frac{100 \times FCC}{FC} \tag{1}$$

OPFCC: We also calculate overall percentage of files that have changed clone code using the following equation.

$$OPFCC_{T_i} = \frac{100 \times \sum_{all\ systems} FCC_{T_i}}{\sum_{all\ systems} FC_{T_i}} \tag{2}$$

Here, T_i represents different types of clones where i = 1, 2 and 3 in all subject systems.

Figure 1 shows the percentage PFCC and the overall percentage OPFCC for seven subject systems individually for each clone type. Here, we can see that Ctags, Camellia and Brlcad have the higher percentage than the rest of the subject systems (jEdit, Freecol, Carol and Jabref). jEdit has the lowest percentage among all. However, in overall percentage we observe that percentage is decreasing in Type 1, Type 2 and Type 3 clone respectively. Table IV describes the FC, FCC and PFCC for all the subject systems individually for each clone type (1, 2 and 3). We can see from this Table that only Camellia has no bug-fix commits related to clone Type 2.

TABLE V
Number of Files that have Changed Non-Clone Code in Bug-fixing Commits

Subject Systems	FNC	FCNC	PFCNC
Ctags	12318	120	0.97%
Camellia	972	49	5.04%
Brlcad	6978	104	1.49%
jEdit	2801	15	0.53%
Freecol	41442	444	1.07%
Carol	13302	94	0.70%
Jabref	39389	333	0.84%

FNC = Number of Files that have Non-Clone code
FCNC = Number of Files that have Changed Non-Clone code
PFCNC = Percentage of the Number of Files that have Changed Non-Clone code

Docking the total number of files containing clone code in bug-fix commits from the total number of files in bug-fix commits we get the total number of files that have non-clone code. For answering RQ 1, we identify the total number of files that have changes in source code. From these files we identify the total number of files that have changes in clone code. The rest of the files made changes to non-clone code due to bug-fix commits.

FNC: This is the number of total files that have non-clone code. Columns with the heading FNC in Table V show the values.

FCNC: This is the number of total files which contain changes in non-clone code. To find out this file number we consider those files which contain changed non-clone code. We also check those files which have changed clone code in addition with non-clone code. All the columns of Table V with the heading FCNC show this value.

PFCNC: To calculate percentage of the number of files containing changed non-clone code we use following equation. All the columns with the heading PFCNC in Table V represent this value. This is shown in equation 3.

$$PFCNC = \frac{100 \times FCNC}{FNC} \tag{3}$$

OPFCNC: We calculate the overall percentage of files containing changed non-clone code using following equation.

$$OPFCNC = \frac{100 \times \sum_{all\ systems} FCNC}{\sum_{all\ systems} FNC} \tag{4}$$

PFCNCs of different clone types for each subject system are shown in Figure 1. Here, we can see that Camellia has

TABLE IV
NUMBER OF FILES THAT HAVE CHANGED CLONE CODE FOUND IN BUG-FIXING COMMITS

	Type 1			Type 2			Type 3		
Subject Systems	**FC**	**FCC**	**PFCC**	**FC**	**FCC**	**PFCC**	**FC**	**FCC**	**PFCC**
Ctags	12	4	33.33%	25	4	16%	117	12	10.25%
Camellia	11	2	18.18%	0	0	0%	56	6	10.71%
Brlcad	33	3	9.09%	5	1	20%	66	5	7.57%
jEdit	21338	117	0.54%	695	14	2.01%	12655	137	1.08%
Freecol	259	13	5.01%	178	11	6.17%	3425	83	2.42%
Carol	121	14	11.57%	138	14	10.14%	991	51	5.14%
Jabref	97	8	8.24%	81	7	8.64%	1143	26	2.27%

FC = Number of Files that have Clone code
FCC = Number of Files that have Changed Clone code
PFCC = Percentage of the Number of Files that have Changed Clone code

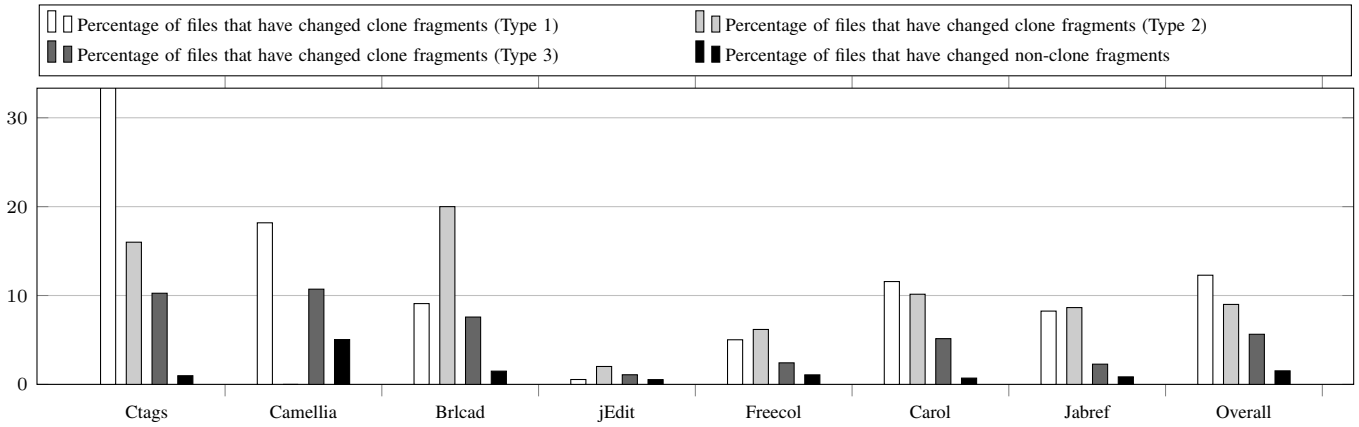


Fig. 1. Percentage of the number of files that have changed clone and non-clone fragments.

the highest percentage than rest of the subject systems. On the other hand, jEdit has the lowest percentage among all subject systems. Table V shows FNC, FCNC and PFCNC for all subject systems. We observe that percentage of changes due to bug-fix commits is higher in clone code than non-clone code. This result was expected because total number of files containing non-clone code is much higher (almost three times) than clone code in every subject system.

The overall percentages (OPFCC) of files that have changes in Type 1 (12.28%) and Type 2 (8.99%) clones have more changes in files than Type 3 (5.63%) clones for bug-fix commits. Moreover, the percentage of files that have changes in clone code is higher than the percentage (OPFCNC) of files that have changes in non-clone code (1.52%).

Mann-Whitney-Wilcoxon (MWW) tests for RQ 1. We are interested to know whether the percentages of three clone types is significantly higher than non-clone code. First, we perform Mann-Whitney-Wilcoxon (MWW) test [34] with percentages of Type 1 clone and non-clone code. We consider the significance level is 5% for this test. According to the data critical U is 13. If the p-value is less than 0.05 and U value is less than 13 then the result is significant. Our result shows that percentage of Type 1 clone code is significantly higher than non-clone code. For two-tailed test we find the p-value of 0.011719 which is much lower than 0.05. In the same way we perform the test for percentages of Type 2 clone code and Type 3 clone code with the non-clone code. We find the p-value of 0.015714 and 0.004574 for Type 2 clone and Type 3 clone respectively. Both percentages of Type 2 clone and Type 3 clone code are significantly higher than non-clone code. Thus, we can say that percentages of all three types of clones are significantly higher than non-clone code. We list our MWW test results in Table VI.

TABLE VI
MANN-WHITNEY-WILCOXON TEST RESULT FOR RQ1

Clone Types	**p-value**	**U value**
Type 1	0.011719	8
Type 2	0.015714	9
Type 3	0.004574	5

Considering level of significance is 5%.
For 5% two-tailed level, Critical value of U is 13

Answer to RQ 1. According to our experimental results percentage of number of file changes in bug-fix commits is higher in clone code than non-clone code. Also, in terms of overall percentage of files Type 1 and Type 2 code clones (12.28% and 8.99%) have higher percentage than Type 3 code clone (5.63%).

We observe that percentages of files containing changes

due to bug-fix commits is high in clone code than non-clone code. However, we still do not know what percentage of clone and non-clone code get changed during bug-fix commits. Intuitively, percentages of bug-fix changes should be higher in clone code than non-clone code. To understand this we investigate our next research question.

B. Answering the second research question (RQ 2)

RQ 2: *How often do bug-fix changes occur to the clone and non-clone code?*

Motivation. Though we have the answer of RQ 1 and hence we know the percentage of files changes in clone and non-clone code but still we are not sure how much these changes are influencing the system. It is important to know the frequency of the bug-fix changes in both clone and non-clone code. From comparison between them we can understand the impact of bug-fix changes. Intuitively, more importance should be given to the more frequent one. This will help us to manage clone code.

Methodology. We know the total number of commits of each subject system. As discussed in Section III-A, we report the total number of commits that have changes in clone fragments. To answer the RQ 2 we automatically count the total number of bug-fix commits that contain changes in clone code. First, we find out the total number of bug-fix commits as described in Section III-B. We automatically count the number of total bug-fix commits which have changed clone code. We deduct this number from the total number of bug-fix commits and found the total number of bug-fix commits which have changed non-clone code. In the following way we calculate the occurrences of bug-fix changes in clone and non-clone code.

CC: This is the total Number of Commits that made changes to Clone code. All the columns of Table VII with the heading CC show this value.

BCC: This is the total Number of Bug-fix Commits that made changes to Clone code. Columns with the heading BCC in Table VII show the values.

PBCC: Percentage of the Bug-fix Commits that made changes to Clone code. We calculate this for each subject systems and for three different types of clone i.e. Type 1, Type 2 and Type 3 clone code. All the columns with the heading PBCC in Table VII represent this value.

We use the following equation for calculating the percentage.

$$PBCC = \frac{100 \times BCC}{CC} \tag{5}$$

We calculate the overall percentage of the bug-fix commits containing clone code using the following equation.

$$OPBCC_{T_i} = \frac{100 \times \sum_{all\ systems} BCC_{T_i}}{\sum_{all\ systems} CC_{T_i}} \tag{6}$$

Here, $OPBCC_{T_i}$ is the overall percentage of the clone code found in the bug-fix commits with respect to T_i type of clones (i=1, 2, 3). Table VII shows the value of CC, BCC and PBCC for seven subject systems and each type of clone individually. Figure 2 describes the percentage PBCCs of all subject systems along with the overall percentage OPBCC. We can see that jEdit has the highest percentage (over 40%) and Brlcad has the lowest percentage (less than 15%). However, in overall percentage Type 3 code clone has the higher percentage than Type 1 and Type 2 code clone.

TABLE VIII
NUMBER OF BUG-FIX COMMITS AFFECTING NON-CLONE CODE

Subject Systems	CNC	BCNC	PBCNC
Ctags	383	137	35.77%
Camellia	133	24	18.04%
Brlcad	589	88	14.94%
jEdit	31	9	29.03%
Freecol	672	326	48.51%
Carol	323	65	20.12%
Jabref	685	161	23.50%

CNC = Number of Commits affecting Non-Clone code
BCNC = Number of Bug-fix Commits affecting Non-Clone code
PBCNC = Percentage of Commits that were applied for fixing Bugs in Non-Clone code

We first identify the list of commits that made changes to the source code. From these commits we detect which commits made changes to clone code. The remaining commits in the list made changes to non-clone code. In the same way we deduct the total number of bug-fix commits containing changes in clone code from the total number of bug-fix commits to find the number of changes in non-clone code bug-fix commits. Applying these findings we answer RQ 2.

CNC: This is the total Number of Commits that made changes to Non-clone code. This value is given in columns with the heading CNC in Table VIII.

BCNC: This is the total Number of Bug-fix Commits that made changes to Non-clone code. Columns with the heading BCNC in Table VIII represent the value.

PBCNC: Percentage of the Bug-fix Commits that made changes to Non-clone code. In Table VIII columns with the heading PBCNC show this value.

Likewise equation 5 to compute the percentages we use equation 7.

$$PBCNC = \frac{100 \times BCNC}{CNC} \tag{7}$$

To see the overall percentage of the number of bug-fix commits that is related to non-clone code we use following equation which is similar to the equation 6.

$$OPBCNC = \frac{100 \times \sum_{all\ systems} BCNC}{\sum_{all\ systems} CNC} \tag{8}$$

Here, $OPBCNC$ is the overall percentage of the bug-fix commits that contain non-clone code. The CNC, BCNC and PBCNC for all subject systems are shown in Table VIII. Here, most of the percentage ranges from 15% to 35% (only the percentage of Freecol has more than 45%). Figure 2 depicts the percentage PBCNCs and overall percentage OPBCNC of

TABLE VII
NUMBER OF BUG-FIX COMMITS AFFECTING CLONE CODE

	Type 1			Type 2			Type 3		
Subject Systems	**CC**	**BCC**	**PBCC**	**CC**	**BCC**	**PBCC**	**CC**	**BCC**	**PBCC**
Ctags	14	3	21.42%	25	4	16%	59	11	18.64%
Camellia	8	1	12.5%	5	1	20%	26	5	19.23%
Brlcad	32	2	6.25%	7	1	14.28%	47	5	10.63%
jEdit	92	37	40.21%	24	10	41.66%	99	42	42.42%
Freecol	35	7	20%	36	10	27.77%	152	46	30.26%
Carol	41	8	19.51%	44	8	18.18%	112	22	19.64%
Jabref	48	6	12.5%	46	6	13.04%	149	23	15.43%

CC = Number of Commits affecting Clone code
BCC = Number of Bug-fix Commits affecting Clone code
PBCC = Percentage of Commits that were applied for fixing Bugs in Clone code

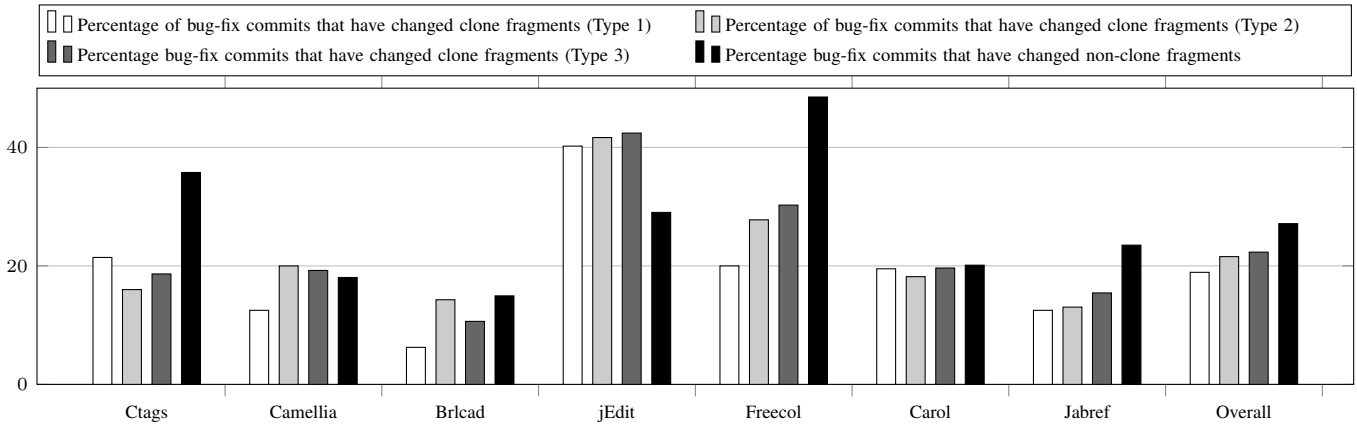


Fig. 2. Percentage of bug-fix commits that have changed clone and non-clone fragments.

seven subject systems. Here, we observe that percentage of the bug-fix commits containing changed non-clone code is highest in Freecol system and lowest in Brlcad. Overall percentage is near 30% which is higher than clone code. Though the bug-fix changes occur more in non-clone code than clone code it is not noteworthy since the difference is not that much high.

The overall percentage (OPBCC) of Type 3 (22.32%) clone is higher than Type 1 (18.91%) and Type 2 (21.56%) clone. Obviously, Type 1 clone (has exact same code) and Type2 clone (has renaming of identifiers or changing data type of identifiers) have less changes than Type 3 clone code (has addition, deletion or modification of code). We believe for this reason Type 3 clone code is affected more than Type 1 and Type 2 clone code. Also, the overall percentage (OPBCNC) for non-clone code (27.13%) is higher than clone code.

Mann-Whitney-Wilcoxon (MWW) tests for RQ 2. We perform MWW test [34] to understand whether the difference between the percentages of clone and non-clone code is significant. Percentage of each type of clone is individually tested with non-clone code. We found the p-value of 0.092892, 0.207578 and 0.344562 for Type 1, Type 2 and Type 3 clone respectively. Calculated U value for Type 1, Type 2 and Type 3 clone code is 16, 20 and 23 respectively. Here, every p-value is greater than 0.05 (significance level is 5%) and every U value is greater than 13 (critical U value is 13). This indicates the result is insignificant. Insignificant result denotes percentage of bug-fix commits having changed non-clone code is slightly higher than clone code. Table IX describes the p-value and U value for all three types of clone code.

TABLE IX
MANN-WHITNEY-WILCOXON TEST RESULT FOR RQ2

Clone Types	**p-value**	**U value**
Type 1	0.092892	16
Type 2	0.207578	20
Type 3	0.344562	23

Considering level of significance is 5%.
For 5% two-tailed level, Critical value of U is 13

Answer to RQ 2. Comparing the overall percentage of bug-fix commits containing clone and non-clone code we found that frequency of bug-fix commits is slightly higher in non-clone code (27.13%) than clone code (20.93%). Also, Type 3 clone code (22.32%) has the higher percentage of changes than Type 1 and 2 clone code (18.91% and 21.56%) due to bug-fix commits.

We observe that bug-fix commits occur in non-clone code more often than clone code by 6.2%. From MWW test we find that the difference between percentages of clone and non-clone code is insignificant.

C. Answering the third research question (RQ 3)

RQ 3: *Is there any difference between the severity of the bugs occurring in clone and non-clone code?*

Motivation. In every single commit there is a message or comment written by the programmer which describes about the changes that they made from the previous commit. In case of bug-fix commits these messages describe about the bug that occur in the code base of the system. By reading a bug-fix commit message we can understand whether a bug is sever or not. This message is helpful for debugging and understanding the scenario of the situation. To understand the severity of bugs and compare between clone and non-clone code we automatically process the bug-fix commit massages followed by a manually inspection. It is important to give priority to more severe bugs while fixing them.

Methodology. We automatically perform a heuristic search proposed by Lamkanfi et al. [35] in the bug-fix commit massages to identify severe bugs. Then we manually investigate the results for validation. We consider five subject systems (Ctags, Camellia, Brlcad, jEdit, Freecol) for this experiment. For the non-clone bug-fix commits we choose some random bug-fix commits which does not contain clone code. It is not feasible to check all the non-clone bug-fix commits by manual inspection. Hence, we keep the total number of non-clone bug-fix commits equal to the total number of clone bug-fix commits to maintain the data impartiality.

Lamkanfi et al. [35] suggested most significant terms for different components indicating severe and non-severe bugs. For example, 'fault', 'hang', 'freez', 'deadlock' etc. represent severe problems of the system. The words 'favicon', 'deprec', 'mnemon', 'outbox' etc. represent non-severe problems of the system. We take these terms as keywords to decide the severity of the bug. Though there are different levels of bug severity we consider only two categories for the simplicity. That is whether the bug is severe or not i. e. 'TRUE' or 'FALSE'. Here, 'TRUE' means the bug is sever (i.e. bug-fix commit messages containing the terms which represent severity) and 'FALSE' means the bug is non-sever (i.e. bug-fix commit messages containing the terms which represent non-severity). This is important for bug triaging process since severe bugs need more care and prompt fixing.

We calculate the percentage of severe bugs in bug-fix commits for both clone and non-clone code. We observe that the existence of severe bugs in Camellia (both clone and non-clone), Brlcad (clone) and jEdit (non-clone) systems is zero (0%). We also observe that Freecol has highest percentage (85.71% for clone and 62.5% for non-clone code) of severe bugs in both clone and non-clone bug-fix commits compared to other subject systems. Percentages of severe bugs in rest of the subject systems are range from 50% to 67%. Ctags (non-clone) and Brlcad (non-clone) have the lowest percentage (50%) of severe bugs. Overall, clone code has higher tendency of having severe bugs than non-clone code. The difference between the overall percentages of severe bugs of clone and non-clone code bug-fix commits is 17.46% which is highly significant. This findings imply that more importance should be given on clone code while fixing bugs for better software maintenance.

> **Answer to RQ 3.** After careful inspection of each commit messages of the bug-fix commits for both clone and non-clone code we found that clone code bug-fix commits has higher percentage of severe bugs (overall percentage 71.63%) than the non-clone code (overall percentage 54.17%). This proves that occurrence of severe bugs is higher in clone code compared with non-clone code.

We observe that severity of bugs is higher in clone code than non-clone code. Though, we find that some of the bug-fix commit messages are very short and it is not enough to describe the severity of the bug. Considering this constraint of the messages in bug-fixing commits the result may vary in different cases. However, severe bugs should have the highest priority in software maintenance.

V. Related Work

Shajnani et al. [24] performed a comparative study between clone and non-clone code for different bug patterns. They used bug reports generated by a tool, i.e., FindBugs whereas we worked on real bug reports that were reported by developers. In our previous study [23] it has been shown that cloning is responsible for replicating bugs. However, it does not show any comparison with non-clone code in that study.

Bug-proneness of code clones has been investigated by a number of existing studies. Li and Ernst [19] performed an empirical study on the bug-proneness of clones by investigating four software systems and developed a tool called CBCD on the basis of their findings. CBCD can detect clones of a given piece of buggy code. Li et al. [36] developed a tool called CP-Miner which is capable of detecting bugs related to inconsistencies in copy-paste activities. Steidl and Göde [20] investigated finding instances of incompletely fixed bugs in near-miss code clones by investigating a broad range of features of such clones involving machine learning. Göde and Koschke [5] investigated the occurrences of unintentional inconsistencies to the code clones of three mature software systems and found that around 14.8% of all changes that occurred to the code clones were unintentionally inconsistent.

None of the studies discussed above investigated bug-fix commits in code clones and non-clone code simultaneously. Mondal et al. [22] investigated bug-proneness of code clones. While the primary target of that study was to compare the bug-proneness of three clone-types, our target is to compare the bug-proneness of clone and non-clone code. Mondal et al. [22] did not investigate the bug-proneness of non-clone code in their study. Focusing on this we perform an in-depth investigation on bug's impacts in code clones and non-clones in this research. Our experimental results are promising and provide useful implications for better understanding of the bug-proneness of clone and non-clone code.

VI. Threats to Validity

We used the NiCad clone detector [25] for detecting clones. While all clone detection tools suffer from the *confounding configuration choice problem* [37] and might give different results for different settings of the tools, the setting that we used for NiCad for this experiment are considered standard [38] and with these settings NiCad can detect clones with high precision and recall [28]–[30]. Thus, we believe that our findings on the bug-proneness of code clones are of significant importance.

Our research involves the detection of bug-fix commits. The way we detect such commits is similar to the technique proposed by Mocus and Votta [33] and also used by Barbour et al. [39]. The technique proposed by Mocus and Votta [33] can sometimes select a non-bug-fix commit as a bug-fix commit mistakenly. However, Barbour et al. [39] showed that this probability is very low. According to their investigation, the technique has an accuracy of 87% in detecting bug-fix commits.

VII. Conclusion

In this paper we conduct an in-depth comparative study of software bugs in both clone and non-clone code. For clone code we also consider three major types of clones, Type 1, Type 2 and Type 3. We investigated thousands of revisions of seven diverse subject systems. We also investigated bug-fix commit messages to measure the frequency of severe bugs in clone and non-clone code. From our examination, changes to files due to bug-fix commits is higher for clone code than for non-clone code. Additionally, changes to files due to bug-fix commits happens more in Type 1 and Type 2 code clones than in Type 3 code clones. In addition, percentage of severe bugs is higher in clone code than non-clone code bug-fix commits. We believe that our findings on bug-fix commits are valuable for better understanding of clone management such as ranking of clone codes and software maintenance.

References

[1] C. K. Roy, M. F. Zibran, and R. Koschke, "The Vision of Software Clone Management: Past, Present, and Future (Keynote paper)," in *Proc. CSMR-WCRE*, 2014, pp. 18–33.

[2] C. K. Roy, "Detection and analysis of near-miss software clones," in *Proc. ICSM*, 2009, pp. 447–450.

[3] L. Aversano, L. Cerulo, and M. D. Penta, "How clones are maintained: An empirical study," in *Proc. CSMR*, 2007, pp. 81–90.

[4] L. Barbour, F. Khomh, and Y. Zou, "Late Propagation in Software Clones," in *Proc. ICSM*, 2011, pp. 273–282.

[5] N. Göde and R. Koschke, "Frequency and risks of changes to clones," in *Proc. ICSE*, 2011, pp. 311–320.

[6] N. Göde and J. Harder, "Clone Stability," in *Proc. CSMR*, 2011, pp. 65–74.

[7] K. Hotta, Y. Sano, Y. Higo, and S. Kusumoto, "Is Duplicate Code More Frequently Modified than Non-duplicate Code in Software Evolution?: An Empirical Study on Open Source Software," in *Proc. EVOL/IWPSE*, 2010, pp. 73–82.

[8] E. Juergens, F. Deissenboeck, B. Hummel, and S. Wagner, "Do Code Clones Matter?" in *Proc. ICSE*, 2009, pp. 485–495.

[9] C. Kapser and M. W. Godfrey, ""Cloning considered harmful" considered harmful: patterns of cloning in software," in *Proc. Empirical Software Engineering*, 2008, pp. 13(6): 645–692.

[10] J. Krinke, "A study of consistent and inconsistent changes to code clones," in *Proc. WCRE*, 2007, pp. 170–178.

[11] J. Krinke, "Is cloned code more stable than non-cloned code?" in *Proc. SCAM*, 2008, pp. 57–66.

[12] J. Krinke, "Is Cloned Code older than Non-Cloned Code?" in *Proc. IWSC*, 2011, pp. 28–33.

[13] A. Lozano and M. Wermelinger, "Tracking clones' imprint," in *Proc. IWSC*, 2010, pp. 65–72.

[14] A. Lozano and M. Wermelinger, "Assessing the effect of clones on changeability," in *Proc. ICSM*, 2008, pp. 227–236.

[15] M. Mondal, C. K. Roy, M. S. Rahman, R. K. Saha, J. Krinke, and K. A. Schneider, "Comparative Stability of Cloned and Non-cloned Code: An Empirical Study," in *Proc. SAC*, 2012, pp. 1227–1234.

[16] M. Mondal, M.S. Rahman, R.K. Saha, C.K. Roy, J. Krinke and K.A. Schneider, "An Empirical Study of the Impacts of Clones in Software Maintenance," in *Proc. ICPC*, 2011, pp. 242-245.

[17] M. Mondal, C. K. Roy, and K. A. Schneider, "An Empirical Study on Clone Stability," in *ACM SIGAPP Applied Computing Review*, 2012, pp. 12(3): 20–36.

[18] S. Thummalapenta, L. Cerulo, L. Aversano, and M. D. Penta, "An empirical study on the maintenance of source code clones," in *Empirical Software Engineering*, 2009, pp. 15(1): 1–34.

[19] J. Li and M. D. Ernst, "CBCD: Cloned Buggy Code Detector," in *Proc. ICSE*, 2012, pp. 310–320.

[20] D. Steidl and N. Göde, "Feature-Based Detection of Bugs in Clones," in *Proc. IWSC*, 2013, pp. 76–82.

[21] L. Jiang, Z. Su, and E. Chiu, "Context-Based Detection of Clone-Related Bugs," in *Proc. ESEC-FSE*, 2007, pp. 55–64.

[22] M. Mondal, C. K. Roy, and K. A. Schneider, "A Comparative Study on the Bug-Proneness of Different Types of Code Clones," in *Proc. ICSME*, 2015, pp. 91–100.

[23] J. F. Islam, M. Mondal, and C. K. Roy, "Bug Replication in Code Clones: An Empirical Study," in *Proc. SANER*, 2016.

[24] H. Sajnani, V. Saini, and C. V. Lopes, "A Comparative Study of Bug Patterns in Java Cloned and Non-cloned Code," in *Proc. SCAM*, 2014, pp. 21–30.

[25] J. R. Cordy and C. K. Roy, "The NiCad Clone Detector," in *Proc. ICPC Tool Demo*, 2011, pp. 219–220.

[26] SVN repository. [Online]. Available: http://sourceforge.net/

[27] M. Mondal, C. K. Roy, and K. A. Schneider, "Spcp-miner: A Tool for Mining Code Clones that are Important for Refactoring or Tracking," in *Proc. SANER*, 2015, pp. 484–488.

[28] C. K. Roy, J. R. Cordy, and R. Koschke, "Comparison and Evaluation of Code Clone Detection Techniques and Tools: A Qualitative Approach," in *Science of Computer Programming*, 2009, pp. 74 (2009): 470–495.

[29] C. K. Roy and J. R. Cordy, "A Mutation / Injection-based Automatic Framework for Evaluating Code Clone Detection Tools," in *Proc. Mutation*, 2009, pp. 157–166.

[30] J. Svajlenko and C. K. Roy, "Evaluating Modern Clone Detection Tools," in *Proc. ICSME*, 2014, pp. 321–330.

[31] C. K. Roy and J. R. Cordy, "A Survey on Software Clone Detection Research," in *Proc. Technical Report 2007-541*, School of Computing, Queen's University, 2007, 115 pp.

[32] 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 *Proc. ICSME*, 2014, pp. 476-480.

[33] A. Mockus and L. G. Votta, "Identifying Reasons for Software Changes using Historic Databases," in *Proc. ICSM*, 2000, pp. 120–130.

[34] Mann-Whitney U Test. [Online]. Available: goo.gl/JIFmo3

[35] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, "Predicting the Severity of a Reported Bug," in *Proc. MSR*, 2010, pp. 1–10.

[36] Z. Li, S. Lu, S. Myagmar, and Y. Zhou, "Cp-miner: A Tool for Finding Copy-paste and Related Bugs in Operating System Code," in *Proc. OSDI*, 2004, pp. 20–20.

[37] T. Wang, M. Harman, Y. Jia, and J. Krinke, "Searching for Better Configurations: A Rigorous Approach to Clone Evaluation," in *Proc. ESEC/SIGSOFT FSE*, 2013, pp. 455–465.

[38] C. K. Roy and J. R. Cordy, "NICAD: Accurate Detection of Near-miss Intentional Clones Using Flexible Pretty-printing and Code Normalization," in *Proc. ICPC*, 2008, pp. 172–181.

[39] L. Barbour, F. Khomh, and Y. Zou, "An empirical study of faults in late propagation clone genealogies," in *Proc. Journal of Software: Evolution and Process*, 2013, pp. 25(11):1139–1165.

The Relationship between Traceable Code Patterns and Code Smells

Zadia Codabux*[1], Kazi Zakia Sultana†[2] and Byron J. Williams‡[2]

[1]Department of Computer Science, Colby College, USA
[2]Department of Computer Science and Engineering, Mississippi State University, USA

Abstract

Context: *It is important to maintain software quality as a software system evolves. Managing code smells in source code contributes towards quality software. While metrics have been used to pinpoint code smells in source code, we present an empirical study on the correlation of code smells with class-level (micro pattern) and method-level (nano-pattern) traceable patterns of code.* ***Objective****: This study explores the relationship between code smells and class-level and method-level structural code constructs.* ***Method****: We extracted micro patterns at the class level and nano-patterns at the method level from three versions of Apache Tomcat and PersonalBlog and Roller from Standford SecuriBench and compared their distributions in code smell versus non-code smell classes and methods.* ***Result****: We found that* $DataManager$*,* $Record$ *and* $Outline$ *micro patterns are more frequent in classes having code smell compared to non-code smell classes in the applications we analyzed.* $localReader$*,* $localWriter$*,* $Switcher$*, and* $ArrReader$ *nano-patterns are more frequent in code smell methods compared to the non-code smell methods.* ***Conclusion****: We conclude that code smells are correlated with both micro and nano-patterns.*

1. Introduction

Software quality is crucial. Detecting and managing code smells improves the quality and maintainability of software systems [10, 12]. A code smell is a surface indication that usually corresponds to a deeper problem in the system [7]. This surface indication may be related to a code or design problem that increases the difficulty of maintenance. [6]. Various raw measures have been derived to assess system characteristics such as lines of code, method length, number of methods etc [11]. These types of measures have been evaluated to establish relative threshold values that are used with other metrics to define code smells. Code smell detection tools are based on the evaluated metrics and established thresholds. A small change in threshold values impacts code smell detection accuracy [5]. Code smells can also be transitively related to each other [6]. Fontana et al. [4] also identified other relations among code smells useful for detecting architectural degradation.

Research has been conducted on code smell detection mechanisms, correlations among code smells, and code smell fault proneness [9, 10, 12, 17, 18]. Fontana et al. analyzed the dependencies between code smells and micro patterns [5]. Micro patterns are class-level traceable patterns that are defined using some formal conditions of the structure of a Java class [8]. Gil et al. [8] developed the concept of traceable patterns and defined 27 micro patterns organized into 8 categories based on the structure of Java classes. These studies were focused on improving software quality by reducing the use of bug-prone micro patterns. Fontana found that dependencies between code smells and micro patterns can be used to improve code quality by detecting code smells using micro pattern data [5]. Method-level traceable patterns are called nano-patterns. Singer et al. [16] listed 17 fundamental nano-patterns organized into 4 groups. They applied their work to clustering and categorizing Java methods based on the associated nano-patterns. Deo et al. [2] found that some nano-patterns such as $localReader$ are highly present in defective methods.

Previous studies focused on finding correlations between code smells and micro patterns. There are certain code smells that are defined based on the methods in a class (e.g., feature envy, long parameter list). In this study, our goal is to determine if correlations exist between code smells with

*zadiacodabux@ieee.org
†ks2190@msstate.edu
‡williams@cse.msstate.edu

DOI reference number: 10.18293/SEKE2017-121

both micro and nano-patterns. These patterns are defined on class or method behavior and code smell occurs due to the violation of fundamental design principles. Therefore, correlating class and method-level structural and behavioral information with code smell will strengthen code smell detection and augment existing smells with pattern-based information. The motivation of this study is to pinpoint code smells in Java classes and methods using their structural information. The patterns having high correlation with code smells can later be used for code smell detection. Moreover, developers will be guided about the use of patterns in code for ensuring better code quality. Detecting code smells in code provides a good indicator of the system's design quality.

The contributions of this study are as follows:

- The results serve to create awareness among developers that using certain patterns can result in particular code smells.
- This study present alternative approaches to identifying code smells and other problematic areas in software.

The paper is organized as follows: Section 2 discusses the related work. Section 3 presents the methodology of our work. Sections 4 presents the results from the experimental analysis and Section 5 discusses the results. Section 6 concludes the paper.

2. Related Work

Prior research focused on finding code smells using various threshold values for related metrics. The problems associated with interpreting these thresholds and setting values applicable to the system under evaluation greatly impact code smell detection accuracy [1, 14]. In order to avoid these problems, researchers examined ways to detect code smells more accurately by correlating smells with other constructs (including other code smells). Fontana [4] extracted relationships among code smells such as determining whether a code smell contains another code smell or if a smelly method calls another smelly method. The authors exploited these relations to tease out architectural anomalies in a system. Other studies described different relationships among code smells [6, 15]. Plain Support and Transitive Support are two other types of code smell relationships [6]. Plain Support denotes the likelihood of the existence of one code smell with the presence of another code smell. On the other hand, Transitive Support is concerned with transitive dependencies among three code smells [6].

There has also been research conducted on the relationships between micro patterns and defects. Destefanis et al. identified certain micro patterns that were more error-prone than others and observed correlations between the patterns and defects [3]. They also showed that the classes that do not contain any micro pattern are more fault prone than classes with micro patterns. In a study by [5], the authors correlated code smells with micro patterns. They explored how structural information can be exploited to detect code smells and bad programming practices. They identified associations among code smells as well as between code smell and micro patterns. Their study is helpful to expedite the code smell detection mechanism as well as to remove discrepancies or contradictions among different code smell detection tools [5].

Although prior studies concentrated on the associations among code smells and micro patterns, in this study we also identify how code smells are correlated to method-level patterns known as nano-patterns. This study results in a new approach for code smell detection using nano-patterns. We also examine correlations between class-level patterns and code smells and compare our results with the previous work.

3. Methodology

This Section elaborates the goal of the study, the research questions, the research approach, and study design.

3.1. Research Goal

The goal of the study is to determine if there exists a correlation between code smells with micro patterns and nano-patterns. This goal is addressed by the research questions presented in the Section 3.2.

3.2. Research Questions

Research Question 1 (RQ1): *How are code smells related to micro patterns?*

The rationale behind this question is to understand if micro patterns are distributed differently in classes having code smells versus classes without code smells. Moreover, we aim to find if there is a correlation between micro patterns with different types of code smells. More specifically:

RQ1.1: *How are micro patterns distributed in classes containing code smells?*

RQ1.2: *Are micro patterns distributed differently in classes with different types of code smells?*

Research Question 2 (RQ2): *How are code smells related to nano-patterns?*

This question is similar to RQ1 except that this analysis is at the method-level as nano-patterns are method-level traceable patterns. More specifically:

RQ2.1: *How are nano-patterns distributed in methods having code smells?*

RQ2.2: *Are the nano-patterns distributed differently in methods with different types of code smells?*

3.3. Study Design

The study was conducted using 3 versions of Apache Tomcat along with PersonalBlog and Roller from Stanford SecuriBench[1]. Apache Tomcat is a web application server developed by the Apache Software Foundation[2]. The software has about a half million lines of code with more than 3000 classes and 12,000 methods in each version. For our study, we considered versions 6.0.45, 7.0.69 and 8.0.33 from the Apache Tomcat Archives[3] (the last released version for each major release during this study). Stanford SecuriBench is a set of Java open source real-life programs. PersonalBlog is a personal blogging application. The software has 38 classes and 275 methods. Roller is a blog server consisting of 226 classes and 2136 methods. In this study, we use the term "code smell classes" to describe the classes where code smells are found and "non-code smell classes" for the classes where no code smell is found.

3.4. Data Extraction

Step 1: Extract code smells: Next, we used Intooitus inCode [4] to extract the code smells from the source code. inCode follows Marinescu's detection strategies to quantify design problems [11] [13]. inCode tests for code smells that are most commonly encountered in software projects. inCode extracts the following specific class-level code smells: god class, data class and schizophrenic class and method-level code smells: Data Clump, Duplication, Feature Envy. We ran inCode on three versions of Apache Tomcat and the two SecuriBench software.

Step 2: Extract micro patterns: The micro pattern tool as described in [8] is interfaced via the command line and accepts a class name or .jar file as input and extracts all micro patterns identified in that class or classes in the jar file. If a particular micro pattern exists in that class, the respective entry is '1' otherwise, it is '0'. We evaluated the unique classes across three versions of Tomcat. For Tomcat, we collected the micro pattern data for a total of 117 "code smell classes" and 7848 "non-code smell classes". For PersonalBlog, we collected data for 9 "code smell classes" and 28 "non-code smell classes". For Roller, we collected data for 33 "code smell classes" and 238 "non-code smell classes".

Step 3: Extract nano-patterns: The nano-pattern detector detects nano-patterns in Java bytecode [16]. We modified the original version of the tool to provide input via a '.properties' file instead of command line arguments and stored the results in a database [2]. For Tomcat, we collected the nano-pattern data for a total 473 "code smell methods" and 37226 "non-code smell methods". For Roller, we collected the nano-pattern data for a total 36 "code smell methods" and 76218 "non-code smell methods".

[1]https://suif.stanford.edu/ livshits/securibench/intro.html

[2]http://tomcat.apache.org/

[3]http://archive.apache.org/dist/tomcat/

[4]https://www.intooitus.com,its evolution at http://www.aireviewer.com

3.5. Data Analysis

Research Question 1 (RQ1): *How are code smells related to micro patterns?*

For RQ1.1, we calculated the percentage of each micro pattern in code smell classes and non-code smell classes. For example, in Tomcat there are 10 classes with the $Record$ micro pattern of the 117 total code smell classes. This micro pattern exists in 8.55% of code smell classes. For RQ1.2, we separated the classes affected by each type of code smell: data class, schizophrenic class and god class. We then computed the percentage of each micro pattern in those classes. For example, in Tomcat we have 18 classes with the $Sink$ micro pattern out of total 35 classes reported to contain the data class type of code smell. Therefore, we can say the percentage of $Sink$ micro pattern in data class code smell classes is 51.4%.

Research Question 2 (RQ2): *How are code smells related to nano-patterns?*

For RQ2.1, we calculated the percentage of each nano-pattern extracted from code smell methods as well as non-code smell methods. For example, in Tomcat there were 192 methods having $ObjCreator$ nano-pattern out of 473 methods having code smells. So 40.6% of the code smell methods have the $ObjCreator$ nano-pattern. Regarding RQ2.2, we separated the methods affected by the following types of code smell: data clumps, duplication and feature envy. The message chain code smell was not a very prominent smell in the system methods and we did not perform any further analysis on it. We then computed the percentage of each nano-pattern in those methods. For example, in Tomcat we have 242 methods having $Void$ nano-patterns out of 300 methods reported to have data clump code smell. Therefore, we can deduce that the percentage of $Void$ nano-pattern in data clump code smell is 80.7%.

For RQ1.1 and RQ2.1, we also performed a Chi-Square Test of Independence to compare the difference in distribution of micro patterns and nano-patterns across the code smell classes and methods and non-code smell classes and methods respectively. We formulated the following null hypotheses with the significance level $\alpha = 0.05$.

$H0$ *The distribution of micro patterns is independent of code smell and non-code smell classes.*

$H0$ *The distribution of nano-patterns is independent of code smell and non-code smell methods.*

We performed separate Chi Square Tests for Independence for the different types of micro patterns and nano-patterns.

4. Results

4.1. Research Question 1 (RQ1)

The test statistic for the chi-Square test of independence involves comparing observed (sample data) and expected frequencies. If the null hypothesis is confirmed, the observed and expected frequencies will be close in value and the chi-Square statistic will be close to zero. If the null hypothesis is false, then the chi-Square statistic will be large. For degrees of freedom of 1 and at 5% levels of significance, the appropriate critical value is 3.84 and the decision rule is as follows: Reject $H0$ if $\tilde{\chi}^2 \geq 3.84$. We computed the chi-square test statistic for each micro pattern and nano-pattern. We reject the null hypothesis when the test statistics for the micro patterns are greater than 3.84. We have statistically significant evidence at $\alpha = 0.05$ to show the distribution of these following micro patterns is not independent of the code smell and non-code smell classes. We have summarized our finding as follows:

- For Apache Tomcat, the micro patterns that are frequent in code smell classes compared to the non-code smell classes are $DataManager$, $Record$ and $Outline$.
- For PersonalBlog, the micro patterns that are frequent in code smell classes compared to the non-code smell classes are $Immutable$ and $Sink$.

Our results on the relations between the different micro patterns and the types of code smells as addressed by RQ1.2 are summarized as follows:
For Apache Tomcat,

- The micro patterns that are more frequent in the classes having data class code smell are $CompoundBox$, $RestrictedCreation$, $CommonState$, $Sink$, $Record$, $FunctionObject$, $Immutable$, $Extender$, $DataManager$, $LimitedSelf$.
- The micro patterns more frequent in the classes having schizophrenic class code smell are $CompoundBox$, $Outline$, $PureType$, $StateMachine$.
- The micro patterns more frequent in the classes having god class code smell are $Implementor$, $Overrider$, $Stateless$, $Recursive$, $LimitedSelf$.

For PersonalBlog,

- The micro patterns that are more frequent in the classes having data class code smell are $Sink$ and $Immutable$.

The Roller application did not contain a significant number of nano-patterns and was excluded from these results.

4.2. Research Question 2 (RQ2)

To address RQ2.1, we separated all the methods containing code smells and then found the nano-pattern distribution in those methods. We calculated the chi-square test statistics for nano-patterns. We reject the null hypothesis when the test statistics for the nano-patterns were greater than 3.84. We have statistically significant evidence at $\alpha = 0.05$ to show the distribution of the nano-patterns is not independent of the code smell and non-code smell methods. We have summarized our findings as follows:

- For Apache Tomcat, the nano-patterns that are more frequent in code smell methods compared to the non-code smell methods are $localWriter$, $Switcher$ and $ArrReader$.
- For Roller, the nano-patterns that are more frequent in code smell methods compared to the non-code smell methods are $localReader$ and $localWriter$.

Our results on the relation between different nano-patterns and the three code smells as addressed by RQ2.2 are summarized as follows:
For Apache Tomcat,

- The nano-patterns more frequent in the methods having data clump code smell are $Void$, $LocalReader$, $JdkClient$ and $TailCaller$.
- The nano-patterns more frequent in the methods having duplication code smell are $LocalReader$, $LocalWriter$, $JdkClient$ and $TailCaller$.
- The nano-patterns more frequent in the methods having feature envy code smell are $objCreator$, $LocalReader$, $LocalWriter$ and $thisInstanceFieldReader$.

For Roller,

- The nano-patterns more frequent in the methods having data clump code smell are $LocalReader$ and $exceptions$.
- The nano-patterns more frequent in the methods having feature envy code smell are $objCreator$ and$LocalWriter$.

The PersonalBlog application did not contain a significant number of nano-patterns and was excluded from these results.

5. Discussion

5.1. Code Smell and Micro Patterns

From Section 4.1, the micro patterns that are frequent in code smell classes compared to the non-code

smell classes are $DataManager$, $Record$ and $Outline$. Fontana [5] found that the $Outline$ micro pattern often results in $SignificantDuplication$ code smell (Confidence level=76%). We also found association of the set of $DataManager$, $Immutable$ and $Extender$ with different code smells at more than 60% confidence.

Data classes are classes which only contain fields, getters / setters, or only public fields. $DataManager$ is a class where all methods are either setters or getters [8]. $Record$ is a class in which all fields are public, no declared methods. $Sink$ is a class whose methods do not propagate calls to any other class. An $Immutable$ class is class whose instance fields are only changed by its constructors. According to 4.1, $DataManager$, $Record$, $Sink$, and $Immutable$ are among the micro patterns that are more frequent in the classes having data class code smell. The definition of these micro patterns support our finding relating to their association with the data class code smell. For example, $Record$ classes contain public fields as data classes do in some cases. As data classes contain only getter or setter methods for finding or setting values to their fields, they do not call other methods to serve any other purpose. Similarly, $Sink$ classes also do not allow its methods to call methods from other classes. Schizophrenic class describes a class with a large and non-cohesive interface. The lack of cohesion is revealed by several disjoint sets of public methods that are used by disjoint sets of client classes. The class with $PureType$ pattern has nothing more than four abstract methods which concrete subclasses must override [8]. $StateMachine$ pattern is an interface to define only parameter-less methods. Such an interface allows client code to either query the state of the object or request the object to change its state in some predefined manner [8]. Therefore, all these types of micro patterns can result in non-cohesive interfaces. $Implementor$ and $Overrider$ micro patterns are among the most frequent micro patterns in the classes containing the god class code smell. God class is an excessively complex class with non-cohesive functionality and heavily manipulates data members from other classes. $Implementor$ is a concrete class, where all the methods override inherited abstract methods. $Overrider$ is a class where all methods override inherited, non-abstract methods [8]. The use of these patterns increases the possibility of having non-cohesive environment.

5.2. Code Smells and nano-patterns

From Section 4.2, the nano-patterns that are more frequent in code smell methods compared to the non-code smell methods are $localWriter$, $Switcher$ and $ArrReader$. $localWriter$ writes values of local variables on the stack frame. $ArrReader$ reads values from an array and $Switcher$ patterns are methods that contain switch statements [16]. We found $ArrReader$ to be a problematic nano-pattern, and this result is in line with the findings of Deo [2] which reported that methods with the $ArrReader$ nano-pattern are highly defect prone.

Data Clumps are large groups of parameters that appear together in the signature of many operations. The nano-patterns that are more frequent in the methods containing the data clump code smell are $Void$, $LocalReader$, $JdkClient$ and $TailCaller$. $Void$ patterns are methods that do not return a value. $LocalReader$ reads values of local variables on stack frame. $JdkClient$ calls methods from the JDK standard library (java.*). $TailCaller$ contains method call followed immediately by return statement [16]. Based on these observations, we can deduce that methods with data clump code smells may not return any value or return control to the calling function after execution. These methods interact and store local variables on the stack. Code Duplication refers to groups of operations which contain identical or slightly adapted code fragments. By breaking the Don't Repeat Yourself (DRY) rule, duplicated code multiplies the maintenance effort, including the management of changes and bug-fixes. Moreover, the code base gets bloated. $LocalReader$, and $LocalWriter$ nano-patterns are among the more frequent patterns in the methods having the duplicated code smell. $LocalWriter$ writes values of local variables on stack frame [16]. $LocalWriter$ has been associated with high defect density [2] and an undesirable effect of code duplication is the introduction of defects. $JdkClient$ may call the same method multiple times and this contributes to code duplication. Feature Envy refers to an operation that is manipulating a lot of data external to its definition scope. In object-oriented code this is a method that uses many data members from a few other classes instead of using the data members of its definition class. $objCreator$ nano-pattern has been found as more frequent in the methods having feature envy code smell. $objCreator$ creates new objects in a method and therefore, its association with a method makes the method more associated with other classes. $objCreator$ has been associated with high defect density as well [2].

6. Conclusion

This study represents preliminary data collection and analysis to determine the relationship of code smells with micro and nano-patterns. We analyzed how the micro patterns and nano-patterns are distributed in code smell and non-code smell classes. Our results can be used for identifying micro patterns and nano-patterns causing code smells. It can also help developers in avoiding the use of particular micro and nano-patterns that often result in code smells. We provided an indication of the most pertinent patterns to code smells by doing a comparative analysis between code smell and non-code smell classes and methods. This study pro-

vides a basis for future work to determine the underlying reason behind the significantly different distribution of patterns in code smell vs non-code smell code. We also plan to extend the study to other systems as well as use these findings to create a prediction model for code smells using traceable patterns.

References

[1] T. L. Alves, C. Ypma, and J. Visser. Deriving metric thresholds from benchmark data. In *Proceedings of the 2010 IEEE International Conference on Software Maintenance*, ICSM '10, pages 1–10. IEEE Computer Society, 2010.

[2] A. Deo and B. J. Williams. Preliminary study on assessing software defects using nano-pattern detection. In *Proceedings of the 24th International Conference on Software Engineering and Data Engineering (SEDE)*, 2015.

[3] G. Destefanis, R. Tonelli, E. Tempero, G. Concas, and M. Marchesi. Micro pattern fault-proneness. In *Proceedings of the 2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, SEAA '12, pages 302–306. IEEE Computer Society, 2012.

[4] F. A. Fontana, V. Ferme, and M. Zanoni. Towards assessing software architecture quality by exploiting code smell relations. In *Proceedings of the Second International Workshop on Software Architecture and Metrics*, SAM '15, pages 1–7. IEEE Press, 2015.

[5] F. A. Fontana, B. Walter, and M. Zanoni. Code smells and micro patterns correlations. In *RefTest 2013 Workshop, co-located event with XP 2013 Conference*, 2013.

[6] F. A. Fontana and M. Zanoni. On investigating code smells correlations. In *Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on*, pages 474–475. IEEE, 2011.

[7] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, 1 edition, July 1999.

[8] J. Y. Gil and I. Maman. Micro patterns in java code. In *Proceedings of the 20th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, OOPSLA '05, pages 97–116. ACM, 2005.

[9] T. Hall, M. Zhang, D. Bowes, and Y. Sun. Some code smells have a significant but small effect on faults. *ACM Trans. Softw. Eng. Methodol.*, 23(4):33:1–33:39, Sept. 2014.

[10] F. Khomh, M. Di Penta, and Y.-G. Gueheneuc. An exploratory study of the impact of code smells on software change-proneness. In *Proceedings of the 2009 16th Working Conference on Reverse Engineering*, WCRE '09, pages 75–84. IEEE Computer Society, 2009.

[11] M. Lanza and R. Marinescu. *Object-oriented metrics in practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems*. Springer Publishing Company, Inc., 2010.

[12] W. Li and R. Shatnawi. An empirical study of the bad smells and class error probability in the post-release object-oriented system evolution. *J. Syst. Softw.*, 80(7):1120–1128, July 2007.

[13] R. Marinescu, G. Ganea, and I. Verebi. Incode: Continuous quality assessment and improvement. In *2010 14th European Conference on Software Maintenance and Reengineering*, pages 274–275, March 2010.

[14] P. Oliveira, M. T. Valente, and F. P. Lima. Extracting relative thresholds for source code metrics. In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week - IEEE Conference on*, pages 254–263, Feb 2014.

[15] B. Pietrzak and B. Walter. Leveraging code smell detection with inter-smell relations. In *Proceedings of the 7th International Conference on Extreme Programming and Agile Processes in Software Engineering*, XP'06, pages 75–84. Springer-Verlag, 2006.

[16] J. Singer, G. Brown, M. Lujn, A. Pocock, and P. Yiapanis. Fundamental nano-patterns to characterize and classify java methods. *Electronic Notes in Theoretical Computer Science*, 253(7):191 – 204, 2010. Proceedings of the Ninth Workshop on Language Descriptions Tools and Applications (LDTA 2009).

[17] A. Yamashita. Assessing the capability of code smells to explain maintenance problems: An empirical study combining quantitative and qualitative data. *Empirical Softw. Engg.*, 19(4):1111–1143, Aug. 2014.

[18] N. Zazworka, M. A. Shaw, F. Shull, and C. Seaman. Investigating the impact of design debt on software quality. In *Proceedings of the 2Nd Workshop on Managing Technical Debt*, MTD '11, pages 17–23. ACM, 2011.

Refactoring Object-Oriented Applications towards a better Decoupling and Instantiation Unanticipation

Soumia Zellagui, Chouki Tibermacine, Hinde Lilia Bouziane, Abdelhak-Djamel Seriai and Christophe Dony

LIRMM, CNRS, University of Montpellier, France
E-mail: {zellagui, tibermacin, bouziane, seriai, dony}@lirmm.fr

Abstract

Modularity in Object-Oriented (OO) applications has been a major concern since the early years of OO programming languages. Migrating existing OO applications to Component-Based (CB) ones can contribute to improve modularity, and therefore maintainability and reuse. In this paper, we propose a method for source code transformation (refactoring) in order to perform this migration. This method enhances decoupling by considering that some dependencies between classes should be set through abstract types (interfaces) like in CB applications. In addition, some anticipated instantiations of these classes "buried" in the source code are extracted and replaced by declarative statements (like connectors in CB applications) which are processed by a dependency injection mechanism. For doing so, a set of "Bad Smells", i.e., modularity-violating symptoms, has been defined. These are first detected in the source code. Then, some refactoring operations are applied for their elimination. An implementation of the method was successfully experimented on a set of open source Java projects. The results of this experimentation are reported in this paper.

1 Introduction

Modularity is a fundamental principle in software engineering, considered as an internal quality attribute that influences external quality attributes, such as maintainability and reuse [7]. A well-modularized system allows collaborative development of different parts (modules) of the same system by different developers. It also enables the substitution or debugging of a module without affecting other modules and the reuse of existing modules in different contexts. Many existing (especially, business) software systems are built using the Object-Oriented (OO) development paradigm. However, many of these systems, especially large ones, are characterized by a high degree of coupling between their elements [13], which makes them difficult to maintain and reuse. Conversly, Component-Based (CB) paradigm has been recognized as an approach that emphasizes software modularity and reuse [4]. Therefore, it would be interesting to migrate OO systems into CB ones. This migration enables to benefit from CB development characteristics, in particular, decoupling and instantiation unanticipation [6].

Existing works that propose migration solutions [2, 3] consider that the modularity and reuse unit, i.e. the component, is a group of classes, called a cluster. In these works, if a user wants to develop a new application by reusing an independent class or subset of classes in a cluster, it is required to reuse the entire component. To the best of our knowledge, no solution proposes refactoring classes individually to make them component descriptors. In this paper, we propose a migration solution that considers each class in an OO application as a component descriptor in a target CB application. This solution follows two steps. The aim of the first step is to detect bad smells (modularity-violating symptoms), i.e. decoupling and unanticipation violation (presented in Section 2). The second step allows the elimination of these bad smells by automatically applying a composition of code refactoring operations (Section 3).

Section 4 discusses the results of an experimentation of this solution conducted on a set of open source Java projects. Section 5 surveys related works and Section 6 concludes the paper and presents future works.

2 Decoupling and Instantiation Unanticipation violation

While code decoupling and instantiation unanticipation principles are fundamental, they are not necessarily always

DOI reference number: 10.18293/SEKE2017-119

respected in existing OO applications. Therefore, it is necessary to identify the symptoms of their violation in an existing OO code to enable their detection and elimintation.

2.1 Decoupling Violation

In CB programming, code decoupling means that components are assumed to communicate only through their interfaces/ports. Therefore, a component has not a direct access to a component with which it interacts. To have this decoupling in OO applications, assuming that each class will correspond to a component descriptor, each class must, for example, expose all its public methods in abstract types (provided interfaces). Then, other classes that use these methods should declare their dependence on these abstract types, which become their required interfaces.

However, most existing OO applications have multiple dependencies between their different classes (direct concrete types) for a cooperative business processing. In particular, it is possible for a field or a parameter to be typed with a concrete class of the application. These situations lead to code decoupling violation.

To deal with decoupling violation, we consider the two bad smells: "Absence or Incompleteness of Provided Interfaces" (AIPI) and "Absence of Required Interfaces" (ARI). AIPI symptoms are identified when: **1)** a class defines a public non-static method[1] not declared in the interfaces implemented by this class (or no interface is implemented by this class), **2)** a class declares public fields or fields with no explicit visibility modifier, and **3)** a class declares global constants. ARI symptoms are identified when a class declares fields with a concrete class type[2].

2.2 Instantiation Unanticipation Violation

In CB applications, instantiation unanticipation means that the implementation of a component does not include a connection to another given component, *i.e.* an instantiation of a class which is another component descriptor. In fact, a component requiring a service can be connected to any other component providing such a service. This connection should be established only by a third party, who is the developer of the application/component that uses the two components to be connected. To comply with this connection fashion in OO applications, constructor calls should not be used. Instead, declarative annotations should be defined; these are processed by a (dependency injection) mechanism that manages instances at runtime.

To deal with instantiation unanticipation violation, we consider the bad smell of type EAI (Existence of Anticipated Instantiations). EAI symptoms are identified when a reference to a created object **1)** is stored in a field/local variable, **2)** is a returned value of a method, or **3)** is an argument of a method invocation. In the present work, we consider that these instantiations are not surrounded by a control flow statement.

The detection of decoupling and instantiation unanticipation violations in an OO application is done by a ("control flow"-insensitive) static analysis of its source code.

[1]Methods of the language's standard API are ignored.

[2]Fields whose types are defined in a library are ignored.

3 Refactoring Operations

This section presents a set of refactoring operations to correct bad smells introduced in the previous section. Table 1 gives an overview of these operations and, for each one, the treated symptom, bad smell.

Table 1: Refactoring Operations

Symptom	Operation
Public fields or fields with no explicit visibility modifier	Change visibilities
Global constants	Move declarations
Public non-static methods not exposed in interfaces	Expose methods
Fields typed with concrete classes	Create required interfaces
Anticipated instantiations	Use dependency injection

Changing the visibility of a field This operation considers the AIPI symptom when a class field is public or has no explicit visibility modifier, i.e., has the package default visibility for Java, for example. In this case, the field visibility is simply changed to private, and a pair of setter/getter methods is inserted to acces this field (only a getter method in the case of a public final field). The resulted methods will be exposed via interfaces as explained through the next refactoring operation.

Exposing Methods through Interfaces

This type of refactoring deals with the AIPI symptom when a class A defines a public non-static method m and its declaration does not exist in any interface. The idea here is to add this declaration to an interface I. This (changed) interface should not be implemented by any other class. Otherwise, i.e., when all the interfaces implemented by A are also implemented by other classes, a new interface I' is created and m's signature is added to it.

We take into consideration the particular case where we distribute the exposed methods on several interfaces. We calculate *LCOM (Lack of Cohesion of Methods)* metric to evaluate the cohesion of each signature added to an interface and the other existing methods in this interface.

Someone can find that the use of *Default Methods* in Java 8 can be useful to eliminate this type of modularity violation. But the idea here consists in exposing only the declarations of methods not their implementations.

Moving a Constant Declaration

To deal with a global constant declaration (AIPI symptom type), we move this declaration to one of the interfaces implemented by the class declaring the constant. We create a new interface or use one from the resulted ones after the application of the previous refactoring operation.

Creating Required Interfaces

This refactoring is used when a field is typed with a concrete class A. It consists of the following steps: search all

invocations to external methods whose receiver is saved in the considered field, collect the signatures of these methods, create a new interface (considered as the required interface), add signatures to this interface and replace the type of the field by the newly created interface. The last step consists of adding inheritance links between the required interface and the provided interfaces implemented by A (the provided interface extends the required interface). The class' required interfaces will be as many as the number of concrete classes used as types for its fields. However a single required interface is created for two fields with the same type. By applying this type of refactoring and the former ones, the required and provided interfaces are henceforth defined explicitly in the source code.

Using Dependency Injection

In Dependency injection (DI), a (client) class does not depend on a specific implementation (concrete class). The implementation class is instantiated and injected at runtime by an object container, such as Spring's one[3].

The first case that we deal with is the one where an instantiation is made inside a method/constructor, the reference to the created instance is stored in a field, and the instantiation does not take any argument or it takes attainable ones (arguments whose values can be calculated by a static analysis). Its refactoring is done through the following steps: save the arguments of the constructor call if any, delete the instantiation statement from the method/constructor body, replace it by an annotation used by the used DI framework. For example, the @Autowired annotation is used on fields in Spring (the field must be non-final). The annotation @Autowired enables the automatic dependency injection based on the type.

The second case that is treated is the one where an instantiation is made inside a method/constructor and the obtained reference is stored in a local variable. As in the previous case, we suppose that the constructor call does not take any argument or take attainable ones. This local variable is removed from the method body and turned into a private field of the class (this refactoring, transforming a local variable to a field, is failure-safe as it has been experimented in the literature [9]). This field will be treated following the previous case. Renaming this local variable, before moving it, could be another additional refactoring.

In contrast to the previous case, since what is transformed is a local variable and not a field, we use here a lazy initialized DI so that the created field is injected when it is first requested (during the execution of the method/constructor where it was originally declared as a local variable), rather than at startup.

The last case is where instances' references are stored in fields/local-variables while using non-literal values as arguments in their instantiation. To deal with this case, first, a new "default" constructor is created in the instantiated class, and the initial constructor call, in the instantiation, is replaced by this new constructor call. Then, a new method that contains exactly what the initial constructor contains is added to the instantiated class. Finally, the instantiation statement is treated following one of the two previous cases, and an invocation statement of the new method is added to the instantiating class.

Anonymous object instantiations, i.e., instantiations which play the role of arguments in method invocations or returned values, for instance, are considered the same as instantiations made as right-hand-side expressions of assignments to local variables. They are processed following the same procedure than the two previous cases.

4 Experiments

We have implemented the described approach[4] using Spoon[5], an open-source library for Java source code analysis and transformation. We conducted some experiments to evaluate the truthfulness of the stated hypothesis of migrating OO applications into CB ones in order to improve their maintainability and reusability quality characteristics. These experiments were conducted to answer the following research questions:

1. What is the efficiency (precision) of the detection phase?
2. To what extent does the proposed approach improve software maintainability?
3. To what extent does the proposed approach improve reusability?

4.1 Used Data & Metrics

For our study, the latest versions of four open source Java projects were used. Table 2 provides a brief description of these projects, which are of different sizes, varying from 5 to 23 KLOC, 50 to 214 concrete types and 1 to 36 abstract types, and developed by different teams to avoid the influence of development team habits on the results.

Table 2: Data collection

System	Description	LOC	No interfaces + Abstract classes	No classes
Jasml-0.10	Java classes visualization tool	5732	1 + 0	50
CoCoME	Commercial application	5779	21 + 0	99
FreeCS-1.3	Chat server	23012	17 + 6	139
Log4j-1.2.17	A Logging API	20129	20 + 16	214

The first research question deals with measuring the efficiency of the detection phase. To answer this question, we measured precision, a well-known metric in information retrieval. Precision assesses the ratio of true smells to all smells detected by our approach. To obtain the set of relevant smells (that should be identified by any smells detection approach), we analysed the four projects manually.

To answer the second research question, we used the *Maitainability Index (MI)* metric that measures the maintainability of a software system, and which was successfully

[3]https://spring.io/

[4]https://cloud.lirmm.fr/index.php/s/QXEV11bUGvYz1Ss

[5]http://spoon.gforge.inria.fr/

Table 3: Detected smells (In each row, M = results of Manual analysis; A = results of Automatic analysis).

System		Public & package fields / All fields	Public non-static methods not exposed / All public non-static methods	Fields of concrete class type	Instantiations that can be injected / All instantiations	Average
Jasml-0.10	M	420/487	48/49	26	41/67	
	A	447/487	49/49	27	46/67	
	Precision	93.96%	97.96%	96.29%	89.13%	94.33%
CoCoME	M	32/285	221/338	25	82/106	
	A	34/285	223/338	29	85/106	
	Precision	94.11%	99.1%	86.20%	96.47%	93.97%
FreeCS-1.3	M	380/888	571/926	83	79/299	
	A	402/888	837/926	88	86/299	
	Precision	94.52%	68.22%	94.31%	91.86%	87.23%
Log4j-1.2.17	M	365/910	750/1015	150	105/246	
	A	370/910	753/1015	167	133/246	
	Precision	98.65%	99.6%	89.82%	78.95%	91.75%

used in many recent works, such as [5]. High MI values indicate that the system is easy to maintain. MI is calculated using the following formula:

*MI1 = 171 - 5.2ln(V) - 0.23*C- 16.2ln(LOC) + (50 * sin(sqrt(2.46*NOLComments))*

V is the Halstead's volume [1], which is a measure of the mental effort required to develop or maintain a program based on its length, number of operators and operands. *C* is the cyclomatic complexity value; *LOC* is the number of lines of code and *NOLComments* is the number of lines of comments. In the case of systems which do not have considerable comments, the above formula can be simplified to omit the involvement of NOLComments. For our study, the tool used to calculate the MI value is JHawk[6].

During software development, programers often reuse existing APIs to write client code. This requires the reuse of all API's classes even if the client application uses only a small fraction of this API. By answering the third research question, we want to validate the assumption that our approach allows shrinking API classes by keeping only the used classes and discarding the other ones, resulting in a reduction of API size in memory. To do this, we used Log4j as an API, on which our approach was applied, and collected four client applications from sourceforge.net that use this API (jdbcLogDriver, VaadingLog4j, Jag and Marauroa), in addition to CoCoME[7] which also uses Log4j. The sizes of these applications range from 8 to 190 classes.

4.2 Protocol & Results

Research Question 1 (efficiency of detection): We asked four master students and one Ph.D. candidate, who were not involved in this work before, to analyse the source code of these systems manually. We gave the Jasml and CoCoMe systems to the two master students. We asked the other two master students to divide the FreeCS system and each of them analyzed half of the packages. The Ph.D. student was assigned to analyse the Log4j system. The five students used, as a reference specification, a detailed description of the bad smells we wrote. They produced Excel files containing the number of occurrences of each bad smell for each class and the total number of smells in the entire project. We report the results of the detection phase in Table 3. It provides for the four systems the number of existing smells, the result of manual analysis (M) in the first line of each row, the smells detected by our implementation (A for automatic) in the second line, and the precision in the third line. Table 3 shows that a large percentage of the results obtained with our approach are validated manually (from 87.23% in average for FreeCS to 94.33% for Jasml). Recall was not measured because in the analyzed applications, there are no false negatives: defaults which are detected manually (relevant) and not detected (retrieved) with our process.

Table 4: MI values before and after applying the refactoring

System	MI before	MI after	Improvement factor
Jasml-0.10	125.49	147.95	1.17
CoCoME	125.12	161.64	1.29
FreeCS-1.3	110.21	120.89	1.09
Log4j-1.2.17	114.52	122.68	1.07

Research Question 2 (improvement in maintainability): We have calculated the aforementioned formula for each class of the analyzed applications. Table 4 shows the MI scores before and after applying the approach on the four projects. MI represents the average of the classes' MI value.

From the results of Table 4, we can observe that MI is improved, with an improvement factor that ranges between 1.07 and 1.29, in the resulting systems after applying the approach. The improvement in maintainability, according to this metric, is not an insignificant score, regarding the size of these systems. Then, in order to check if during the evolution of a single system, the proposed refactorings keep stable this improvement in maintainability, we evaluated MI for six versions of Log4j API, which where developed over a period of 17 years.

The MI values for the six versions before and after applying the proposed refactorings are depicted in Table 5. From this table, we can see that there is an increase in MI values of Log4j, before applying our approach, in all the analyzed versions. This is justified by the fact that from a version to another, new functionalities are added to the system or bugs are corrected, but developers of this system pay attention to its maintainability. As an example of modifications that have been performed in version 1.0.4 and contributed

[6]http://www.virtualmachinery.com/index.htm

[7]http://www.cocome.org/

to improve the maintainability of version 1.1.3: *FileAppender* class from *org.apache.log4j* package has been splitted into three classes (*ConsoleAppender*, *WriterAppender* and *FileAppender*) and the MI value for this class passed from 120.47 to 122.75 (the average MI for the three new classes).

Table 5: MI values of Log4j versions

Version	# Classes	MI before	MI after	Imp. factor
1.0.4	146	111.31	121.59	1.09
1.1.3	162	112.25	122.47	1.09
1.2.1	179	114.81	120.66	1.05
2.0	87	116.08	119.64	1.03
2.4	112	117.66	121.56	1.03
2.8	172	118	121.16	1.02

As we can observe, in all the versions, the maintainability is improved. However, the improvement is greater in the first versions. This is explained by the fact that starting from the (major) version 2.0, the structure of Log4j has completely changed, and its maintanability was substantially improved. In the following versions, the system keeps a good MI score, even if this is slightly improved by our refactorings. This shows that our refactorings give better results on old legacy systems, compared to new, potentially refactored, ones.

In the following paragraphs, we report on a case study we have conducted in order to evaluate the benefits brought by the proposed approach on the maintenance effort in API migration. API migration is a kind of software adaptation, and is part of the software maintenace activities. The effort in API migration is measured in this case study in terms of the number of tokens in the modified lines of code in a client application. The two APIs (source and target) of our study are XOM [8] and JDOM [9] which are XML document manipulation APIs. We performed an API migration, from XOM to JDOM, of a client application named SleepXomXML [10]. Measurements have been made before and after applying our approach on this application and for the two APIs.

Table 6: API migration results

	SleepXOMXML		Refactored SleepXOMXML	
	XOM	JDOM	Refact. XOM	Refact. JDOM
Number of tokens in modified lines	254	600	269	380
Difference	346		111	

The results of this case study are shown in Table 6. In this table, we can see the number of tokens in the lines of code that have been adapted in the version of the client application, before its refactoring using our approach: 254 tokens. The number of these tokens in the new code (after its migration to JDOM) become 600 (third column). (Difference = 346 tokens.) When considering the application after its refactoring with our approach, the number of tokens in the modified lines is slightly more, 269. But the number of tokens in the new code became much less, 380. We can deduce from the table that in this case study, the difference in the number of tokens has been reduced by more than 3 times (from 346 to 111).

Research Question 3 (improvement in reusability:) To determine which classes are really used by the Log4j client applications, we analysed these applications's source code manually. Table 7 shows the result of this analysis.

The first column presents the number of classes/interfaces used directly in the client source code. Both *CoCoME* and *jdbcLogDriver* use only one class which is *Logger*. *VaadingLog4j* uses three classes (*Category*, *Priority* and *LoggingEvent*) and *Marauroa* uses two interfaces (*Appender* and *LoggerRepository*) and seven classes. These directly used classes/interfaces have dependencies with other classes/interfaces of Log4j. Their number is depicted in the second column. The proportion of the API used code is 70% for *jdbcLogDriver, VaadingLog4j, Marauroa* and *CoCoME* (the same classes/interfaces are used by these client apps). *Jag* uses directly only one class, *LogLevel*. This is explained by the fact that *Jag* uses another logging implementation which is Apache Commons Logging.

In terms of memory usage, the Log4j API size is 1.3 MB. By applying our approach and keeping only the used classes/interfaces by the client applications, this size can be reduced to 0.7 MB for *jdbcLogDriver, VaadingLog4j, Marauroa* and *CoCoME* and 8KB for *Jag*.

4.3 Threats to validity

The obtained results in the detection phase depend on the specification of the bad smells and on the profile of students. We tried to be as accurate as possible in the description of smells and we have chosen students who have some experience in Java programming. Another aspect can biais the results and is related to the number of persons involved in the experiments: one student was assigned to one system or to a part of a system. Several persons should be assigned per system to have more accurate results. To mitigate this risk, we gave these students large periods of time (2 weeks in average) to carefully check the smells; and asked them to analyse each class individually and indicate the time spent on that class. The individual results can be checked in the previous repository.

Besides, we tried to collect systems of different sizes and developped by different teams to diversify the data. It is sure that with a larger set of systems we may obtain more precise results. However, since the results were all positive with the four studied systems, which vary in size, our intuition, on the interest of transforming OO code into CB one using the proposed refactoring operations, is strengthened.

5 Related Works

Allier et al [2] proposed a method to automate the process of migrating OO Java applications into CB OSGi ones. This method makes component interfaces operational by the use of two design patterns: Adapter and Façade. In another

[8] http://www.xom.nu/

[9] http://www.jdom.org/

[10] http://altsol.gr/sleepxomxml/

Table 7: Number of used classes/interfaces in Log4j

Client applications	API's directly used types (classes + interfaces/abstract classes)	API's indirectly used types (classes + interfaces/abstract classes)	API (classes + interfaces/abstract classes)
jdbcLogDriver	1	145 + 28	214 + 36
VaadingLog4j	3	143 + 28	
Jag	1	1	
Marauroa	7 + 2	137 + 28	
CoCoME	1	145 + 28	

work, Alshara et al [3] proposed another approach to automatically transform Java applications into OSGi ones. This approach takes as input a Java application and the description of its CB architecture. Then, the code transformation consists of replacing the dependencies (inheritance and instantiation) between classes belonging to different clusters (components) by interactions via interfaces. Shatnawi et al [12] proposed an approach that aims at recovering software components from OO APIs. In this approach, groups of API classes that are able to form components are identified. This identification is based on the probability of classes to be reused together by clients, and the structural and behavioral dependencies among classes.

In these works, the modularity unit, and therefore the reusability unit, is a group of classes (a cluster). If a user wants to develop a new application using an independent class or a subset of classes in a cluster, she/he is obliged to reuse the entire component. To optimize the level of reuse, we defend here the idea of refactoring the classes individually, to make them component descriptors.

In [8], Fowler defined 22 refactorings for Java programs and initially introduced the concept of bad smells in code as an indicator when (and where) to apply refactorings. Shah et al [11] proposed an algorithm that uses various refactoring techniques to automatically remove unwanted dependencies in Java programs. This algorithm is designed to eliminate modularity defects represented by four types of anti-patterns: circular dependencies between packages, subtypes knowledge, abstraction without decoupling and degenerated inheritance. They classified dependencies between classes in four categories and for each category they specified a refactoring operation. For decoupling classes using interfaces, Steimann et al [13] proposed a fully automated refactoring approach for the introduction of new interfaces. This refactoring calculates from variable decalarations, the minimal types (interfaces), containing all the method declarations needed from the chosen reference and all other references it gets possibly assigned to.

These works share the same goal, which is improving the modularity of an application. Our method has the same goal but with another requirement which is having at the end of the process a class that complies with a component descriptor, in which the decoupling is "pushed further", through the declaration of dependencies as abstract types only, and *via* instantiation unanticipation.

6 Conclusion and Futur Work

In this paper, we presented a method for improving the modularity of object-oriented source code, by focusing on what component-based development brought to programming, *i.e.* decoupling and instantiation unanticipation. Our method was experimented on a set of Java projects to evaluate its efficiency in the detection of modularity violations, and the improvement it brings to maintainability and reusablity. The results of this experimentation showed that there is a potential in using the proposed process in migrating existing legacy OO applications.

As a future work, we plan to perform our analysis on a larger set of applications (with larger sizes). In addition, we envisage to take into consideration other OO mechanisms, like inheritance (by replacing inheritance by delegation like in [10]) and instantiation in nested classes, which bring new instances of bad smells in the analyzed projects. From a tool-support point of view, we project to integrate our solution to the Eclipse IDE as a monolithic refactoring operation and experiment its usability.

References

[1] R. E. Al Qutaish and A. Abran. An analysis of the design and definitions of halstead metrics. In *IWSM*, 2005.

[2] S. Allier et al. From object-oriented applications to component-oriented applications via component-oriented architecture. In *WICSA*, 2011.

[3] Z. Alshara et al. Migrating large object-oriented applications into component-based ones: instantiation and inheritance transformation. In *GPCE*, 2015.

[4] A. Bertolino et al. An architecture-centric approach for producing quality systems. In *Quality of Software Architectures and Software Quality*. Springer, 2005.

[5] J. Börstler et al. Beauty and the beast: on the readability of object-oriented example programs. *Software Quality Journal*, 2016.

[6] L. Fabresse et al. Foundations of a simple and unified component-oriented language. *Computer Languages, Systems & Structures*, 34(2):130–149, 2008.

[7] N. E. Fenton et al. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Co., 1998.

[8] M. Fowler. *Refactoring - Improving the Design of Existing Code*. Addison-Wesley, 1999.

[9] M. Gligoric et al. Systematic testing of refactoring engines on real software projects. In *ECOOP*. Springer, 2013.

[10] H. Kegel et al. Systematically refactoring inheritance to delegation in java. In *ICSE*. IEEE, 2008.

[11] S. M. A. Shah et al. On the automation of dependency-breaking refactorings in java. In *ICSM*, 2013.

[12] A. Shatnawi et al. Reverse engineering reusable software components from object-oriented apis. *JSS*, 2016.

[13] F. Steimann et al. Decoupling classes with inferred interfaces. In *SAC*, 2006.

Delphi: A Source-code Analysis and Manipulation System for Bricklayer

Victor Winter
Department of Computer Science
University of Nebraska-Omaha
Omaha, Nebraska
vwinter@unomaha.edu

Betty Love
Department of Mathematics
University of Nebraska-Omaha
Omaha, Nebraska
blove@unomaha.edu

Chris Harris
Department of Computer Science
University of Nebraska-Omaha
Omaha, Nebraska
charris@unomaha.edu

Abstract

Delphi is a source-code analysis and manipulation system being developed to analyze and transform Bricklayer programs. The information obtained from Delphi analysis can be used to generate problem-specific text in the form of a mini-lecture. This opens the door to the automated integration of such texts with commercial animation software and text-to-speech (TTS) tools. The result is a scalable infrastructure capable of providing formative feedback to students in the form of an animated cartoon whose information is personalized (e.g. male/female actors, use of slang and dialects) and problem-specific. This feedback can be provided in a timely fashion and can ease technical burdens on educators that teach coding across the K-12 spectrum.

1 Introduction

This paper debuts *Delphi*, a source-code analysis system we are developing for analyzing Bricklayer programs. Within the Delphi infrastructure, it is possible to perform a wide range of customized analysis. The information obtained from such analysis can be used to generate problem-specific text in the form of a mini-lecture. This opens the door to the integration of such texts with commercial animation software and text-to-speech tools. The result is a system which provides *formative feedback* to students in the form of an animated cartoon whose information is personalized. Formative feedback is defined as "information communicated to the learner that is intended to modify his or her thinking or behavior to improve learning" [2].

The rest of the paper is organized as follows: Section 2 gives an overview of related work. Sections 3 and 4 give brief overviews of the Bricklayer ecosystem and Delphi respectively. Section 5 gives an example showing how Delphi can be used to provide animated formative feedback for pixel art artifacts constructed in Bricklayer. Section 6 looks towards the future and describes the kinds of analysis that is possible using Delphi, and Section 7 concludes.

DOI reference number: 10.18293/SEKE2017-010

2 Related Work

A literature review paper written by Keuning et. al focuses on tools that provide automated feedback for programming exercises [1]. They report on 69 feedback generation tools. A finding of their review is that tools (1) generally do not "give feedback on fixing problems and taking a next step", and (2) cannot easily be adapted to specific needs of teachers. Delphi can address both of these concerns.

3 Overview of the Bricklayer Ecosystem

Bricklayer [5] is an online educational ecosystem designed in accordance with a "low-threshold infinite-ceiling" philosophy. Its purpose is to teach coding to people of all ages and coding backgrounds. A significant portion of the Bricklayer ecosystem has been developed specifically to help novices, especially primary school children, learn how to code. When executed, Bricklayer programs can produce LEGO® artifacts, Minecraft artifacts, as well as artifacts suitable for 3D printing. Bricklayer resides in a domain in which there is a strong connection between math, art, and computer science. The Bricklayer ecosystem is freely-available and can be found at:

`http://bricklayer.org`

Bricklayer programs are written in the functional programming language SML. Graphical capabilities are provided by the Bricklayer library. The output of Bricklayer programs are files which are input to third-party tools which include: LEGO® Digital Designer (LDD), LDraw, Minecraft, and STL viewers such as 3D Builder.

Bricklayer programs can also be developed using a block-based editor called *bricklayer-lite*. Bricklayer-lite is built using Google Blockly. A noteworthy capability of bricklayer-lite is that, in addition to producing a Bricklayer (graphical) artifact, the execution of a bricklayer-lite program will produce a well-formed and well-formatted Bricklayer program text which can be executed outside of the bricklayer-lite framework.

4 Delphi

Delphi is a source-code analysis tool that is being developed for Bricklayer. From an implementation standpoint, Delphi represents a non-trivial extension of the TL system (a general-purpose program transformation system) specialized to the domain of Bricklayer. The TL system [6][3] is a powerful and mature meta-programming system which has been used to develop tools for a variety of domains including: (1) a source-code analysis system for Java, (2) a compiler for an architecture independent microprogramming language, (3) compiler optimizations, and (4) automated test generators.

One of the primary design goals of Delphi is to provide a tool facilitating specification of custom analysis rules (e.g., domain specific or even problem specific analysis rules).

The basic capabilities of Delphi include dependency analysis as well as standard syntax-driven metrics such as source lines of code (SLOC) as well as metrics relating to the use of various programming language constructs (e.g., conditional expressions and curried function declarations).

5 An Example of Animated Formative Feedback

A very engaging and artistic activity in Bricklayer involves the creation of pixel art – an example of which is shown in Figure 1. This activity centers on the translation of pixel art images into Bricklayer code. Pixel art projects can be individual or group oriented and the choices of what pixel art image to code are enormous. Pixel art images found on the web range from very simple to extremely complex.

Given an assignment to "code up" a pixel art image, a student has a wide range of options. This enables the selection of images that have special interest or meaning to the coder. The selection process is also influenced by an assessment of the complexity of the image as well as a self assessment of the ability of the individual.

It is not uncommon for a student to employ a *greedy algorithm* when coding a pixel art artifact.

DOI reference number: 10.18293/SEKE2017-010

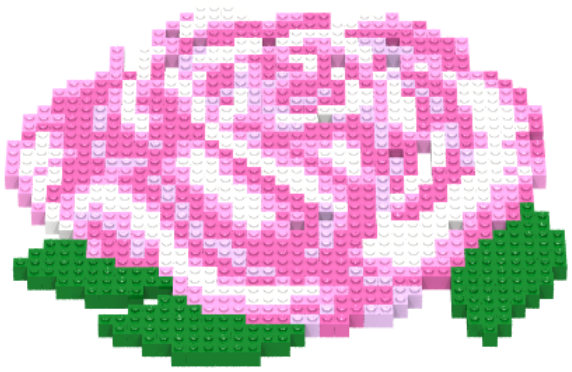


Figure 1: An example of a pixel art artifact created by an elementary school student.

When using such an algorithm one strives to place as many large bricks as possible and then places smaller bricks and so on. This approach can minimize the number of function calls in the source code. However, our experiences in working with student coding pixel art have led us to believe that the complexity of the boundary between the portion of the pixel art image that has been completed and the portion that remains to be completed becomes intellectually unmanageable when using a greedy algorithm. Though our results are anecdotal in nature, we have found that, due to its high cognitive load, a greedy algorithm for coding pixel art is fairly unreliable and, in the end, extremely labor intensive.

5.1 A Row-Major Algorithm

The most effective way to code pixel art is "one row/-column at a time". You code a row, run the code, validate that the row you coded is correct, and then move on and code the next row. We refer to such a construction algorithm as a *row-major algorithm*.

Bricklayer currently has five coding levels. Each level has greater expressive powers than the level that precedes it. Coding levels 1 and 2 only provide a standard set of brick shapes and colors for coding in the xz-plane. In particular, when subjected to the constraints imposed by a row-major algorithm, only 2×1 and 1×1 bricks are available for use. In levels 1 through 4, such bricks are placed using *put* function calls.

In order to be *minimal*, a program (i.e., code) implementing a row-major algorithm must satisfy a number of properties. First, the sequence of coordinates as which bricks are placed must satisfy a row-major order. Let (x_i, z_i) and (x_j, z_j) denote the coordinates of two put function calls, where a brick is placed at (x_i, z_i) before (x_j, z_j). To satisfy row-major order, $z_i < z_j \vee (z_i = z_j \wedge x_i < x_j)$.

A minimal program may not contain any brick *collisions*. A collision occurs when two or more put function calls place a brick in the same cell. In this case, Bricklayer's default behavior is to overwrite the contents of the cell according to the most recent put function call (i.e., the last brick you place is what you will see).

A minimal program should not be *locally compressible*. For example, two put function calls that place 1×1 blue bricks adjacent to one another should be replaced by a single put function call that places a 2×1 blue brick.

5.2 About Delphi Analysis

Through the use of 22 rules, Delphi is able to transform any input program p_{in} that creates a 2-dimensional artifact into a corresponding output program p_{out} such that (1) p_{out} constructs the 2-dimensional artifact according to a row-major algorithm, and (2) p_{out} is minimal according to the definition given above. During this program transformation, Delphi also records, in a database, which transformation rules are used as well as how they are used. The information stored in this database is then used to generate a natural language report describing the modifications that need to be performed in order to make the p_{in} compliant with a row-major algorithm. Figure 2 describes two of the rules used by Delphi.

Delphi is able to analyze and transform the contents of file hierarchies (e.g., hundreds of Bricklayer programs) at the touch of a single button. For each program that is processed a resulting text can be pro-

DOI reference number: 10.18293/SEKE2017-010

Condition	Action	Rationale
1. Two put function calls place bricks of the same color at adjacent coordinates (e.g., (0,0) and (1,0)). 2. The first put function call places a 2×1 brick and the second put function call places a 1×1 brick.	The second put function call can be removed.	The 1×1 brick placed by the second function call will overwrite the brick placed by the first put function call. However, since both put functions place the same color brick no visible change will occur as a result of the second put function call. Therefore, the second put function call can be removed without changing the appearance of the artifact.
1. In a sequence consisting of three put function calls, the first two put function calls place bricks of different colors at the same coordinate. Let (x, z) denote this coordinate. 2. The first put function call places a 2×1 brick, and the second put function call places a 1×1 brick. 3. The third put function call places a 2×1 brick at the coordinate $(x + 2, z)$.	Modify the first put function call so that it places a 1x1 brick at the coordinate $(x + 1, z)$, and then lexically commute the first and second put function calls (i.e., position the modified (previously) first function call after the second function call).	The first put function call places a 2×1 brick which will occupy cells (x, z) and $(x + 1, z)$. In contrast, second put function call places a 1×1 brick which will only occupy the cell (x, z). Since the bricks placed by the first two put function calls have different colors the bricks at (x, z) and $(x + 1, z)$ will have different colors. Furthermore, the third put function call places a 2×1 brick so its shape cannot be increased. Thus, the first put function call must be modified so that is places a 1×1 brick at $(x + 1, z)$. The first and second put function calls must then be commuted in order for the coordinates associated with put function calls to conform to a row-major order.

Figure 2: Descriptions of some rules used by Delphi.

duced explaining, at a level deemed appropriate for the author of the code, the changes that would need to be made in order to place the Bricklayer program under consideration into row-major form. This text can then be given as input to a text-to-speech (TTS) translator and the result can be given as input to an animation program.

Figure 3 shows a screenshot of a video created in CrazyTalk in the manner just described: The dialog was produced automatically using Delphi. This was then copy-and-pasted into the text-to-speech (TTS) translator for CrazyTalk. The animations were hand made, but used canned motions provided by CrazyTalk. The images of the Bricklayer program were constructed by hand as was the rendering of the artifact. Though the creation of this demonstration involved manual steps, in the future, animation tools, such as CrazyTalk, could be parameterized on such inputs. The result would be a system where the production of such custom videos would be completely automatic.

DOI reference number: 10.18293/SEKE2017-010

Watch the video at: `https://www.youtube.com/watch?v=o6aoW5rgOhU&feature=youtu.be`

A sample of a dialog generated by Delphi

The artifact created by the Bricklayer program in the file called panda.bl is not constructed according to a row-major algorithm. Not to worry, I will tell you how you can change your program so that it conforms to a row-major algorithm. The 1-by-1 brick created by the function call on line number 78 is overwritten by the function call that follows it. Therefore, it can be removed without changing the appearance of your artifact. Your program contains a pair of function calls that put a 2-by-1 brick and a 1-by-1 brick, having different colors, at coordinates whose x-value differ by 1. The function-call at line number 85 can be changed so that it puts a 1-by-1 brick instead of a 2-by-1 brick.

Figure 3: Animated problem-specific feedback provided by Delphi.

6 Future Work

One of the primary design goals of Bricklayer was to create a coding domain that is "example rich and problem dense" [4]. In such a domain, the creation of smooth learning curves become possible which can be supported through numerous examples and similar problems/exercises.

The learning curve in Bricklayer is, in part, realized through a sequence of concepts. In many cases the transition from one concept to the next can be understood in equational terms. This is by design and is facilitated by the underlying semantics of functional programming languages like SML (the language in which Bricklayer programs are written) in which referential transparency play a central role. Consequently, this also provides a domain in which a program transformation system, like Delphi, can play

DOI reference number: 10.18293/SEKE2017-010

an important role.

In traditional textbooks, new concepts are introduced using a *static* set of examples. Examples that were created when the textbook was published. With Delphi, it is possible to create examples *dynamically*. For example, a student program containing no user-defined functions can be transformed automatically by Delphi into an equivalent program containing user-defined function declarations and calls. This results in a "living textbook" in which learning examples are (transformationally) derived from individual programs created by students. Furthermore, it is also possible for such transformations to generate text explaining what was done. Such texts can then be fed into animation systems to produce personalized learning material.

7 Conclusion

We believe that educational technology has just scratched the surface regarding what is possible. Technologies lying within our near-term reach can be developed to provide significant technical support for educators interested in teaching coding. Technologies, such as Delphi, when added to the support services provided by online help desks can address teacher needs in effective and cost-efficient ways. Such a model also geographically decouples technical expertise from teaching expertise. One help desk can serve the educational needs of schools around the world. We believe that such approaches will be needed to solve the problem of teaching coding across the K-12 spectrum.

References

[1] H. Keuning, J. Jeuring, and B. Heeren. Towards a systematic review of automated feedback generation for programming exercises. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '16, pages 41–46, New York, NY, USA, 2016. ACM.

[2] V. J. Shute. Focus on Formative Feedback. *Review of Educational Research*, 78(1):153–189, 2008.

[3] The TL System, 2010. `http://faculty.ist.unomaha.edu/winter/ShiftLab/TL_web/TL_index.html`.

[4] V. Winter. Bricklayer: An Authentic Introduction to the Functional Programming Language SML. *Electronic Proceedings in Theoretical Computer Science (EPTCS)*, 2014.

[5] V. Winter. The world needs more computer science! what to do? In D. Conway, S. A. Hillen, M. Landis, M. T. Schlegelmilch, and P. Wolcott, editors, *Digital Media, Tools, and Approaches in Teaching and Their Added Value*, pages 119–141. Waxmann Verlag GmbH, Germany, 2015.

[6] V. L. Winter. Stack-based Strategic Control. In *Preproceedings of the Seventh International Workshop on Reduction Strategies in Rewriting and Programming*, June 2007.

DOI reference number: 10.18293/SEKE2017-010

A Simpler and More Direct Derivation of System Reliability Using Markov Chain Usage Models

Lan Lin, Yufeng Xue
Department of Computer Science
Ball State University
Robert Bell Building, Room 455
Muncie, IN 47306, USA
{llin4, yxue2}@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 been around for more than two decades, and 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 reviews the analytical derivation of the single use reliability mean, and proposes a simpler, faster, and more direct way to compute the expected value that renders an intuitive explanation. The new derivation is illustrated with two examples.

1 Introduction

Statistical testing based on a *Markov chain usage model*, as a rigorous testing method developed by the University of Tennessee Software Quality Research Laboratory (UTK SQRL), has been around for more than two decades [6, 5, 10, 8, 11, 14, 16, 15] and successfully applied in a variety of industry and government projects, ranging from medical devices to automotive components to scientific instrumentation, to name a few [3, 2, 13]. The direct benefit statistical testing provides is a quantitative analysis of the system's quality using empirical data that can be used to demonstrate, document, and certify that the system is fit for its intended use.

In statistical testing a Markov chain usage model is first developed. States of the Markov chain represent states of system use (such as "Cruise Control Activated," "Cruise Control Deactivated," or "Cruise Control Resumed" in an automobile cruise control system). Arcs between states represent possible transitions between states of use (such as "Activating Cruise Control" when the driver presses a button on the steering wheel, or "Deactivating Cruise Control" when the driver engages the brake). Each arc has an associated probability of making that particular transition, based on expected usage data in the field. The outgoing arcs from each state have probabilities that sum to one. Test cases can be generated from the model by different sampling options. Pass and fail data are recorded and analyzed for reliability estimation, coverage analysis, or stopping decisions. This form of statistical testing [6, 5, 10, 8, 11, 14, 16, 15] supports quantitative certification of software by a statistical protocol. A public domain tool supporting statistical testing (*JUMBL*: *J Usage Model Builder Library* developed by UTK SQRL) is freely available [9, 1].

This paper reviews the current reliability analysis for statistical testing based on a Markov chain usage model, and specifically how to derive a system end-to-end reliability estimate based on the testing experience given the usage model, as established in [10, 12], and proposes a simpler way to compute the expected value of system reliability, not via the system failure probability (or unreliability), but directly. The new derivation results in a simpler closed-form formula with less steps of computation that also renders an intuitive explanation. We illustrate it through two examples and compare the results with those obtained from the old analysis.

The remainder of the paper is organized as follows. Section 2 discusses related work and compares different reliability models that have been used for statistical testing. Section 3 reviews the current analytical derivation of the system reliability. Section 4 presents our new formula with two illustrating examples. Finally Section 5 concludes the paper and points out directions for future work.

DOI reference number: 10.18293/SEKE2017-091

2 Related Work: Reliability Analysis

After a sample of test cases are generated from the usage model and executed, results of testing are recorded. A test case is considered successful if all the steps (events) constituting the test case are executed successfully; otherwise, the steps on which a test case fails are recorded, together with the information as to whether testing continues or stops after each failure step. The recorded testing results are empirical data used to estimate the system reliability.

Several reliability models that have been used for statistical testing are presented and compared in [10, 12]. They are briefly described below:

- *The sampling model* [7]. Each executed test case is a Bernoulli trial. An estimated reliability is computed at a confidence level, based on the numbers of successful and unsuccessful test cases. Variations among tests are not taken into account, with long and short tests equally treated.

- *The failure state model* [16]. The testing result is stored as a testing (Markov) chain. It contains states from the usage model that have been visited during testing, and additional failure states for observed failures. The failure states are made absorbing. The upper bound on system reliability is defined as the probability of going from the source to the sink of the testing chain without being absorbed in a failure state. The model takes into account the probability mass of each test case, however, in the absence of failures (no absorbing states) the estimate of reliability is one.

- *The Bayesian model* [4]. Miller et al. presented a reliability model based on Bayesian theory. It allows for the use of prior (testing) information, and assumes a standard beta distribution for the software failure probability (or "unreliability," that is, the probability of a randomly chosen use failing). An expected value for system reliability and the associated variance are computed from a posterior beta distribution, based on both prior and observed successes and failures.

- *The arc-based Bayesian model* [10, 12]. The Miller model is applied to individual arcs of a Markov chain usage model. For each arc the expected value of the arc failure probability and the associated variance are computed from a posterior beta distribution. A *single use reliability* is defined as "the probability of the software executing a randomly selected use without a failure." This model takes into account variations among tests, makes use of prior testing information, yields a reliability estimate based on the amount of testing when testing reveals no failures, and provides a variance estimate.

The arc-based Bayesian approach is appealing because first, it takes into account variations among tests induced by the model structure; second, it provides insight into where lack of testing or unreliability is having a greater impact on the system level end-to-end reliability, as a failure on one arc may not have the same impact as a failure on a different arc; and third, it makes it easier for the tester to estimate the pre-test reliability for a single arc, which contributes to a more accurate estimate of the system reliability. For these reasons the latest version of the JUMBL [1] adopts the arc-based Bayesian model.

3 Single Use Reliability and Its Derivation

Using the arc-based Bayesian model one could compute the system end-to-end reliability through the *single use reliability* estimate, either via analytical derivation [10] or through simulation [12]. Single use reliability is defined as "the probability of a randomly selected use executing correctly relative to a specification of correct behavior [6, 10]." Since the analytical solution in closed form [10] is both faster and more precise than simulation, it was implemented in the latest JUMBL tool. We summarize the major steps and results of the derivation below.

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 [12], 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. The prior success and failure counts are estimated from prior testing experience or knowledge obtained from code analysis, design records, previous failure data, etc. In case no prior information is available, $a_{i,j} = b_{i,j} = 1$. The observed success and failure counts are collected through testing. 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 of $r_{i,j}$:

$E[r_{i,j}] = \frac{\alpha_{i,j}}{\alpha_{i,j}+\beta_{i,j}} = \frac{a_{i,j}+s_{i,j}}{a_{i,j}+s_{i,j}+b_{i,j}+f_{i,j}}$.

Given $f_{i,j} = 1 - r_{i,j}$, we can compute the mean of $f_{i,j}$ as $E[f_{i,j}] = E[1 - r_{i,j}] = 1 - E[r_{i,j}]$.

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, and F_1), 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 failures, 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 $\dot{R}_1$ denote the submatrix of R_1 omitting the last column.

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.

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 two entry-wise products as follows. One is of size $(n-1) \times n$: $\mathscr{F}_1 = Q \otimes F_1$. The other is a square matrix of order $n-1$: $\dot{\mathscr{R}}_1 = \dot{Q} \otimes \dot{R}_1$.

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 [10] that F^* in (1) computes the expected *single use failure probability* (or *single use unreliability*) from any starting state.

$$F^* = (I - \dot{\mathscr{R}}_1)^{-1} \mathscr{F}_1 U \quad (1)$$

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 (1).

For an intuitive understanding of (1), 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 [10] that $(I - \dot{\mathscr{R}}_1)^{-1} = \dot{\mathscr{R}}_1^0 + \dot{\mathscr{R}}_1^1 + \dot{\mathscr{R}}_1^2 + \ldots$, 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 $\mathscr{F}_1$ 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.

A more complex equation is also given in [10] for computing the variance associated with the single use reliability from any starting state, which we are not elaborating here due to the page limits.

To sum up, the following steps are required to compute the system reliability mean as illustrated in [10]:

1. Determine Q and $\dot{Q}$ from the usage model.
2. Determine A and B from prior success and failure counts for each arc in the usage model.
3. Determine S and F from observed success and failure counts for each arc in the usage model.
4. Compute R_1 and F_1 from A, B, S, and F.
5. Compute $\dot{\mathscr{R}}_1$ and $\mathscr{F}_1$.
6. Compute F^* by (1).
7. The expected value of the single use reliability is one minus the first component of F^*.

4 A Simpler and More Direct Derivation of the Single Use Reliability Mean

We think the derivation of the single use reliability mean (expected value) could be simplified, and not through the single use failure probability (or single use unreliability) but directly, as shown below.

4.1 The New Derivation

We define another entry-wise product of size $(n-1) \times n$: $\mathscr{R}_1 = Q \otimes R_1$. Let W be $\mathscr{R}_1$ restricted to the last column. $\mathscr{R}_1$ is a column vector of size $n-1$.

We define R^* as follows:

$$R^* = (I - \dot{\mathscr{R}}_1)^{-1} W \quad (2)$$

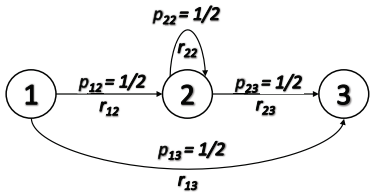


Figure 1. Example 1 of a Markov chain usage model. Arcs are annotated with transitional probabilities and arc reliabilities.

(2) has an intuitive explanation. As explained above for (1), 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 $\mathscr{R}_1$ 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 (2).

We demonstrate below two examples of computing the single use reliability mean by the formula in [10] (hereafter referred to as Stacy and Poore's formula) as well as by our new formula. For both examples the new formula gets to the same result with fewer steps and simplified calculation.

4.2 Examples

Figure 1 illustrates our first example of a Markov chain usage model. The arcs are annotated with transitional probabilities and arc reliabilities.

By Stacy and Poore's formula:

$$P=\begin{bmatrix}0&\frac{1}{2}&\frac{1}{2}\\0&\frac{1}{2}&\frac{1}{2}\\0&0&0\end{bmatrix}\quad Q=\begin{bmatrix}0&\frac{1}{2}&\frac{1}{2}\\0&\frac{1}{2}&\frac{1}{2}\end{bmatrix}\quad \dot{Q}=\begin{bmatrix}0&\frac{1}{2}\\0&\frac{1}{2}\end{bmatrix}$$

$$R_1=\begin{bmatrix}r_{11}&r_{12}&r_{13}\\r_{21}&r_{22}&r_{23}\end{bmatrix}\quad \dot{R}_1=\begin{bmatrix}r_{11}&r_{12}\\r_{21}&r_{22}\end{bmatrix}\quad I=\begin{bmatrix}1&0\\0&1\end{bmatrix}$$

$$\dot{\mathscr{R}}_1=\dot{Q}\otimes\dot{R}_1=\begin{bmatrix}0&\frac{r_{12}}{2}\\0&\frac{r_{22}}{2}\end{bmatrix}\quad I-\dot{\mathscr{R}}_1=\begin{bmatrix}1&-\frac{r_{12}}{2}\\0&1-\frac{r_{22}}{2}\end{bmatrix}$$

$$(I-\dot{\mathscr{R}}_1)^{-1}=\frac{1}{1-\frac{r_{22}}{2}}\begin{bmatrix}1-\frac{r_{22}}{2}&\frac{r_{12}}{2}\\0&1\end{bmatrix}=\begin{bmatrix}1&\frac{\frac{r_{12}}{2}}{1-\frac{r_{22}}{2}}\\0&\frac{1}{1-\frac{r_{22}}{2}}\end{bmatrix}$$

$$F_1=1-R_1=\begin{bmatrix}1-r_{11}&1-r_{12}&1-r_{13}\\1-r_{21}&1-r_{22}&1-r_{23}\end{bmatrix}$$

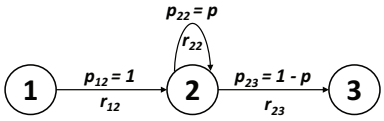


Figure 2. Example 2 of a Markov chain usage model. Arcs are annotated with transitional probabilities and arc reliabilities.

$$\mathscr{F}_1=Q\otimes F_1=\begin{bmatrix}0&\frac{1-r_{12}}{2}&\frac{1-r_{13}}{2}\\0&\frac{1-r_{22}}{2}&\frac{1-r_{23}}{2}\end{bmatrix}\quad U=\begin{bmatrix}1\\1\\1\end{bmatrix}$$

$$(I-\dot{\mathscr{R}}_1)^{-1}\mathscr{F}_1=\begin{bmatrix}0&\frac{1-r_{12}}{2}+\frac{\frac{r_{12}}{4}(1-r_{22})}{1-\frac{r_{22}}{2}}&\frac{1-r_{13}}{2}+\frac{\frac{r_{12}}{4}(1-r_{23})}{1-\frac{r_{22}}{2}}\\0&\frac{\frac{1-r_{22}}{2}}{1-\frac{r_{22}}{2}}&\frac{\frac{1-r_{23}}{2}}{1-\frac{r_{22}}{2}}\end{bmatrix}$$

$$F^*=(I-\dot{\mathscr{R}}_1)^{-1}\mathscr{F}_1U=\begin{bmatrix}\frac{1-\frac{r_{22}}{2}-\frac{r_{13}}{2}+\frac{r_{13}r_{22}}{4}-\frac{r_{12}r_{23}}{4}}{1-\frac{r_{22}}{2}}\\\frac{1-\frac{r_{22}}{2}-\frac{r_{23}}{2}}{1-\frac{r_{22}}{2}}\end{bmatrix}$$

$$R^*=1-F^*=\begin{bmatrix}\frac{\frac{r_{13}}{2}-\frac{r_{13}r_{22}}{4}+\frac{r_{12}r_{23}}{4}}{1-\frac{r_{22}}{2}}\\\frac{\frac{r_{23}}{2}}{1-\frac{r_{22}}{2}}\end{bmatrix}$$

By our new formula:

$$W=\begin{bmatrix}\frac{r_{13}}{2}\\\frac{r_{23}}{2}\end{bmatrix}\quad R^*=(I-\dot{\mathscr{R}}_1)^{-1}W=\begin{bmatrix}\frac{r_{13}}{2}+\frac{\frac{r_{12}r_{23}}{4}}{1-\frac{r_{22}}{2}}\\\frac{\frac{r_{23}}{2}}{1-\frac{r_{22}}{2}}\end{bmatrix}$$

$$=\begin{bmatrix}\frac{\frac{r_{13}}{2}-\frac{r_{13}r_{22}}{4}+\frac{r_{12}r_{23}}{4}}{1-\frac{r_{22}}{2}}\\\frac{\frac{r_{23}}{2}}{1-\frac{r_{22}}{2}}\end{bmatrix}$$

We may also compute the single use reliability (SUR) directly based on its definition given the model structure. 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:

$$SUR=\sum_{i=0}^{\infty}(\tfrac{1}{2}\cdot(\tfrac{1}{2})^i\cdot\tfrac{1}{2}\cdot r_{12}\cdot r_{22}^i\cdot r_{23})+\frac{r_{13}}{2}=\frac{r_{12}r_{23}}{4}\sum_{i=0}^{\infty}(\frac{r_{22}}{2})^i+\frac{r_{13}}{2}=\frac{\frac{r_{12}r_{23}}{4}}{1-\frac{r_{22}}{2}}+\frac{r_{13}}{2}=\frac{\frac{r_{13}}{2}-\frac{r_{13}r_{22}}{4}+\frac{r_{12}r_{23}}{4}}{1-\frac{r_{22}}{2}}$$

Figure 2 illustrates our second example of a Markov chain usage model.

By Stacy and Poore's formula:

$$P=\begin{bmatrix}0&1&0\\0&p&1-p\\0&0&0\end{bmatrix}\quad Q=\begin{bmatrix}0&1&0\\0&p&1-p\end{bmatrix}\quad \dot{Q}=\begin{bmatrix}0&1\\0&p\end{bmatrix}$$

$$R_1=\begin{bmatrix}r_{11}&r_{12}&r_{13}\\r_{21}&r_{22}&r_{23}\end{bmatrix}\quad \dot{R}_1=\begin{bmatrix}r_{11}&r_{12}\\r_{21}&r_{22}\end{bmatrix}\quad I=\begin{bmatrix}1&0\\0&1\end{bmatrix}$$

$$\dot{\mathscr{R}}_1=\dot{Q}\otimes\dot{R}_1=\begin{bmatrix}0&r_{12}\\0&r_{22}p\end{bmatrix}\quad I-\dot{\mathscr{R}}_1=\begin{bmatrix}1&-r_{12}\\0&1-r_{22}p\end{bmatrix}$$

$$(I-\dot{\mathscr{R}}_1)^{-1} = \frac{1}{1-r_{22}p}\begin{bmatrix} 1-r_{22}p & r_{12} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & \frac{r_{12}}{1-r_{22}p} \\ 0 & \frac{1}{1-r_{22}p} \end{bmatrix}$$

$$F_1 = 1 - R_1 = \begin{bmatrix} 1-r_{11} & 1-r_{12} & 1-r_{13} \\ 1-r_{21} & 1-r_{22} & 1-r_{23} \end{bmatrix}$$

$$\mathscr{F}_1 = Q \otimes F_1 = \begin{bmatrix} 0 & 1-r_{12} & 0 \\ 0 & p(1-r_{22}) & (1-p)(1-r_{23}) \end{bmatrix}$$

$$(I-\dot{\mathscr{R}}_1)^{-1}\mathscr{F}_1 = \begin{bmatrix} 0 & 1-r_{12}+\frac{p(1-r_{22})r_{12}}{1-r_{22}p} & \frac{(1-p)(1-r_{23})r_{12}}{1-r_{22}p} \\ 0 & \frac{p(1-r_{22})}{1-r_{22}p} & \frac{(1-p)(1-r_{23})}{1-r_{22}p} \end{bmatrix}$$

$$U = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad F^* = (I-\dot{\mathscr{R}}_1)^{-1}\mathscr{F}_1 U = \begin{bmatrix} \frac{1-r_{22}p-r_{12}r_{23}+r_{12}r_{23}p}{1-r_{22}p} \\ \frac{1-r_{22}p-r_{23}+r_{23}p}{1-r_{22}p} \end{bmatrix}$$

$$R^* = 1 - F^* = \begin{bmatrix} \frac{r_{12}r_{23}-r_{12}r_{23}p}{1-r_{22}p} \\ \frac{r_{23}-r_{23}p}{1-r_{22}p} \end{bmatrix}$$

By our new formula:

$$W = \begin{bmatrix} 0 \\ (1-p)r_{23} \end{bmatrix} \quad R^* = (I-\dot{\mathscr{R}}_1)^{-1}W =$$

$$\begin{bmatrix} 1 & \frac{r_{12}}{1-r_{22}p} \\ 0 & \frac{1}{1-r_{22}p} \end{bmatrix}\begin{bmatrix} 0 \\ (1-p)r_{23} \end{bmatrix} = \begin{bmatrix} \frac{r_{12}r_{23}-r_{12}r_{23}p}{1-r_{22}p} \\ \frac{r_{23}-r_{23}p}{1-r_{22}p} \end{bmatrix}$$

By the definition of single use reliability:

$$SUR = \sum_{i=0}^{\infty} 1 \cdot p^i(1-p)\cdot r_{12}\cdot r_{22}^i \cdot r_{23} = \sum_{i=0}^{\infty} r_{12}r_{23}(1-p)(pr_{22})^i = \frac{r_{12}r_{23}(1-p)}{1-pr_{22}} = \frac{r_{12}r_{23}-r_{12}r_{23}p}{1-r_{22}p}$$

5 Conclusion and Future Work

Statistical testing based on a Markov chain usage model has been well established in theory and proved sound and effective in practice [6, 5, 10, 8, 11, 14, 16, 15], with tools available to support all the stages of testing and to automate the testing process [1, 9]. This paper presents a simpler way to compute the system end-to-end reliability mean, not through the system failure probability (or unreliability) but directly, following the arc-based Bayesian model [10, 12]. We illustrate it with two examples and compare the results with those obtained from the old analysis.

Work is under way to implement the new formula in the supporting tool JUMBL, and to examine the computation of the single use reliability variance to find an equivalent but simpler, more direct, and more intuitive derivation as well (the current one is complex and counter-intuitive).

Acknowledgements

This work was generously funded by Ontario Systems through the NSF Security and Software Engineering Research Center (S^{2}ERC).

References

[1] 2017. J Usage Model Builder Library (JUMBL). Software Quality Research Laboratory, The University of Tennessee. http://jumbl.sourceforge.net/jumblTop.html.

[2] T. Bauer, T. Beletski, F. Boehr, R. Eschbach, D. Landmann, and J. Poore. From requirements to statistical testing of embedded systems. In *Proceedings of the 4th International Workshop on Software Engineering for Automotive Systems*, pages 3–9, Minneapolis, MN, 2007.

[3] L. Bouwmeester, G. H. Broadfoot, and P. J. Hopcroft. Compliance test framework. In *Proceedings of the 2nd Workshop on Model-Based Testing in Practice*, pages 97–106, Enscede, The Netherlands, 2009.

[4] 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.

[5] J. H. Poore. Theory-practice-tools for automated statistical testing. *DoD Software Tech News: Model-Driven Development*, 12(4):20–24, 2010.

[6] 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.

[7] J. H. Poore, H. D. Mills, and D. Mutchler. Planning and certifying software system reliability. *IEEE Software*, 10(1):88–99, 1993.

[8] 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.

[9] 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.

[10] 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.

[11] S. J. Prowell, C. J. Trammell, R. C. Linger, and J. H. Poore. *Cleanroom Software Engineering: Technology and Process*. Addison-Wesley, Reading, MA, 1999.

[12] 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.

[13] K. Sayre and J. H. Poore. Automated testing of generic computational science libraries. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, page 277c, Big Island, HI, 2007.

[14] G. H. Walton, J. H. Poore, and C. J. Trammell. Statistical testing of software based on a usage model. *Software – Practice & Experience*, 25(1):97–108, 1995.

[15] J. A. Whittaker and J. H. Poore. Markov analysis of software specifications. *ACM Transactions on Software Engineering and Methodology*, 2(1):93–106, 1993.

[16] 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 Process to Calculate the Uncertainty of Software Metrics-based Models Using Bayesian Networks

Renata M. Saraiva*[1], Mirko Perkusich†[1], Hyggo Almeida‡[1], and Angelo Perkusich§[1]

[1]Embedded and Pervasive Computing Laboratory, Federal University of Campina Grande, Campina Grande, Brazil

Abstract

Software metrics are essential resources in software enterprises. They can be used to support decision-making and, consequently, reduce costs, improve the productivity of the team and the quality of products delivered. On the other hand, this is only possible if the metrics are valid. Although there are studies related to software metrics validity, none present a solution to represent the uncertainties of the metrics selected to measure the attributes of the entities. In this paper, we present a process to build Bayesian networks to represent the uncertainties of software metrics-based models. The proposed solution is composed of two activities and focuses on the selection and validation of metrics to construct the Bayesian networks. We validated the model with simulated scenarios. Given the successful results, we concluded that the proposed solution is promising. This paper complements the state of the art by showing how to complement a popular metric selection technique, GQM, with information to model uncertainties of the metrics using the concepts of metric validation and Bayesian networks.

Software metrics selection; Software metrics validation; Goal-Question-Metric; Validation criteria; Bayesian networks.

1 Introduction

According to Finkelstein and Leaning [9], measurement is the objective representation of an empirical knowledge of a real-world entity. According to Mathias et al [13], a measurement occurs when an attribute is measured, that is, when a value is assigned to it. By combining this measure with useful information, we have a metric (e.g., average number of defects per module). In practice, the terms "metric" and "measure" are often used interchangeably [13]. In this paper, we use the definitions presented by Mathias et al. [13]. However, we consider that each measure is a metric and this, in turn, may be composed of more than one measures.

There are many applications in the field of software metrics such as quality assessment and prediction. For instance, Quamoco [19] focuses on measuring product quality and Hearty et al. [11], on predicting the velocity of an XP project.

Despite the benefits of using metrics and the various researches on software metrics undertaken in recent years, the acceptance and use of metrics in practice is still an ongoing concern: more than 80% of software measurement initiatives fail within the first 18 months. One possible explanation for this phenomenon is the difficulty to understand and use metrics [20].

According to Fenton and Neil [7], metrics have been used successfully to quantify, but they have not been properly used to support decision-making. A reason for the limited adoption with this purpose is the lack of trustworthiness on the validity of metrics. For instance, Chidamber-Kemerer (CK) metrics are popularly used to evaluate Object Oriented-based software. On the other hand, Kitchenham [12] discusses that two of the proposed metrics (Lack of Cohesion and Coupling Between Objects) are theoretically invalid. In other words, they do not represent the attributes of the entities in which they were proposed to. Using invalid metrics result in meaningless (i.e., totally arbitrary) decisions.

There are studies that propose criteria to evaluate the validity of software metrics. In Meneely et al. [14], results

*renata.saraiva@embedded.ufcg.edu.br
†mirko.perkusich@embedded.ufcg.edu.br
‡hyggo@embedded.ufcg.edu.br
§perkusic@embedded.ufcg.edu.br

DOI reference number: 10.18293/SEKE2017-172

of a systematic literature review are presented, in which 47 criteria were identified. On the other hand, there are no proposed solutions to, given an attribute of an entity that needs to be measured, calculate how representative (i.e., valid) is the set of metrics selected to measure it. In the context of Goal Question Metric (GQM), a popular software metrics paradigm, there is no solution to model the uncertainty of the set of metrics used to answer a question using the criteria presented in Meneely et al. [14].

In this paper, we present a process to build Bayesian networks to represent the uncertainties of software metrics-based models. The process is composed of two activities: (i) metrics selection and (ii) metrics validation. The first activity is composed of three steps: characterization of the environment, acquisition of knowledge through abstraction sheets and construction of the Bayesian network. The second activity is composed of two steps: execution of the validation method from validation criteria and update the Bayesian network.

We used Bayesian networks because they are flexible to be learned from data or elicited from domain experts. Since metric models can be applied in contexts in which there are historical data and in which there are not, this flexibility is crucial. Furthermore, it can deal with different types of data (e.g., discrete, continuous, Boolean and ordinal), which adds flexibility to the types of metrics to be used. Finally, it deals with uncertainty and enables the modeling of cause-consequence relationships, which enables the modeling of the validity of metrics and build GQM-based models.

To validate our solution, we used ten simulated scenarios. Based on the results, we concluded that it is a promising approach to assist on the construction of interpretation-oriented metric programs. We plan to complement our process with threshold definition techniques [17] and metrics reliability activities [15]. This paper complements the state of the art by showing how to complement a popular metric selection technique, GQM, with information to model uncertainties of the metrics using the concepts of metric validation and Bayesian networks.

This paper is organized as follows. Section 2 presents an overview on Bayesian networks. Section 3 presents our proposed solution. Section 4 presents our validation and Section 5 presents our final remarks.

2 Bayesian Networks

Bayesian networks are probabilistic graph models used to represent knowledge about an uncertain domain [2]. A *Bayesian network*, N, is a directed acyclic graph that represents a joint probability distribution over a set of random variables V [10]. The network is defined by the pair $N = \{G, \Theta\}$. G is the directed acyclic graph in which the nodes $X_1, \ldots, X_n$ represent random variables and the arcs represent the direct dependencies between these variables. Θ represents the set of the probability functions. This set contains the parameter $\theta_{x_i|\pi_i} = P_N(x_i|\pi_i)$ for each x_i in X_i conditioned by π_i, the set of the parameters of X_i in G. Equation 1 presents the joint distribution defined by N over V.

$$P_N(X_1, \ldots, X_n) = \prod_{i=1}^{n} P_N(x_i|\pi_i) = \prod_{i=1}^{n} \theta_{X_i|\pi i} \qquad (1)$$

Bayesian networks have many advantages such as suitability for small and incomplete data sets, structural learning possibility, combination of different sources of knowledge, explicit treatment of uncertainty, support for decision analysis, and fast responses [18]. Furthermore, they can combine the knowledge of domain experts and historical data to build more realistic models in an approach called smart-data [4]. To construct the Bayesian networks presented in this study we used AgenaRisk[1].

This technique has been applied to build software metrics-based models for several purposes in software engineering such as risk management [6], product quality management [19], effort prediction [11] and process management [15].

To reduce the effort of defining the Node Probability Tables (NPTs) through elicitation of knowledge from domain experts, Fenton et al. [8] proposed the concept of ranked nodes, which is based on the doubly truncated Normal distribution (TNormal) limited in the $[0, 1]$ region. We used ranked nodes because the goal is to give meaning to the metric. Therefore, we used an ordinal scale. An advantage of ranked nodes, when compared to other approaches to define NPT for ordinal variables, is the explicit configuration of the confidence in the result (i.e., variance).

3 Proposed Process

The goal of the proposed solution is to represent the uncertainties of software metrics-based models. For this purpose, we used Bayesian networks. It is composed of two activities: (i) metrics selection and (ii) metrics validation. The first activity is composed of three steps: characterization of the environment, acquisition of knowledge through abstraction sheets and construction of the Bayesian network. The second activity is composed of two steps: execution of the validation method from validation criteria and update the Bayesian network. In Figure 1, we present an activity diagram representing the process.

3.1 Software Metrics Selection

To select the metrics, we use the GQM [1] paradigm. First, it is necessary to identify the project context (i.e.,

[1] http://www.agenarisk.com/

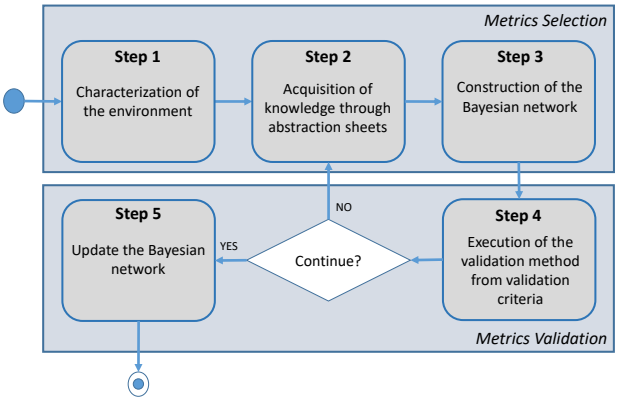


Figure 1. Process overview.

application domain and development process) from the domain experts. For instance, semi-structured interviews with project leaders might be performed.

With the context defined, the next step is to apply the GQM process with the goal of capturing the experience of the viewpoints and selecting the software metrics. For this purpose, abstraction sheets must be used as knowledge acquisition instrument during interviews [3].

A set of goals is defined as $G = \{g_1, \ldots, g_{|G|}\}$, where g_i represents a project goal. For each goal, an abstraction sheet, which is composed of four quadrants, must be used. A set of questions, Q, and metrics, M, must be extracted from the first quadrant, which corresponds to the quality focus. $q_i \in Q$ and $m_i \in M$ represent, respectively, a question and a metric related to a project goal.

The second quadrant corresponds to the variation factors, which are factors that impact the quality focus, considering the defined goal. More questions q and metrics m can be extracted from this quadrant.

The third quadrant corresponds to the possible values of the extracted metrics from the first quadrant. These values are important because they demonstrate the usefulness of the measurement process. By analyzing them, we can detect discrepancies between expectations and realities.

Finally, the last quadrant of the abstraction sheet corresponds to the impact of the variation factors on the extracted metrics from the first quadrant. The description of this impact serves as motivation for the inclusion of the variation factor in the abstraction sheet. If the project leader does not know how to inform the impact of a variation factor, this factor should be excluded from the sheet.

Given that each question must be associated with at least one goal, a set of relationships between goals and questions, T, should be created. $t(g, q)$ means the goal g and question q are related. An hierarchical structure example of the GQM model is illustrated in Figure 2.

For the example shown in Figure 2, the given goal is to "Analyze the software product with respect to its quality for the purpose of characterization from the developer's point of view". For quality focus, the question "How many unwanted behaviors does the product have?" was defined. For the variation factor, the question "What is the quality of the test?" was defined. An example of an abstraction sheet is illustrated in Figure 3.

By analyzing first quadrant of the abstraction sheet presented in Figure 3, it is possible to identify metrics such as *number of detected failures*, *proportion of critical/uncritical failures* and *number of detected faults*. Given that the goal, questions and metrics for the quality of focus are defined, the Bayesian network can be built. For our approach, all node should be modelled as ranked. If a metric is collected using a numerical scale, thresholds must be defined to convert it into an ordinal scale. For this purpose, statistics-based approach [17] can be used or data must be collected from domain experts.

After the construction of the directed acyclic graph, the NPT of the goal node must be defined. Assuming that the goal was modeled as a ranked node, we can create a truth table to collect data from a domain expert and define the NPT. For the given example, given that there is a one-to-one relationship between g and q, the truth table is not necessary and the NPT must be calibrated as an identity matrix, in which the diagonal elements are 1 and the remaining are 0.

3.2 Software Metrics Validation

The validation of the metrics ensures that they are representative of the measured attributes. In the literature, there are many researches on the validation of software metrics [16, 12, 14]. Meneely et al. [14] performed a systematic review about validation criteria for software metrics and identified 47 criteria. In Table 1, we present ten criteria identified by Meneely et al. [14].

A priori validity	*Monotonicity*
Actionability	*Metric Reliability*
Appropriate Continuity	*Non-collinearity*
Appropriate Granularity	*Non-exploitability*
Association	*Non-uniformity*

Table 1. List of 10 validation criteria found in the review [14].

Defining the purpose of using a metric is a critical step to validate it. In addition, when the project leader makes a decision, he can specify properties of the metrics that are most appropriate to use. According to Meneely et al. [14] this is called advantage. For instance, be able to show that a metric is a significant representation and that it can be applied to a development process are considered advantages.

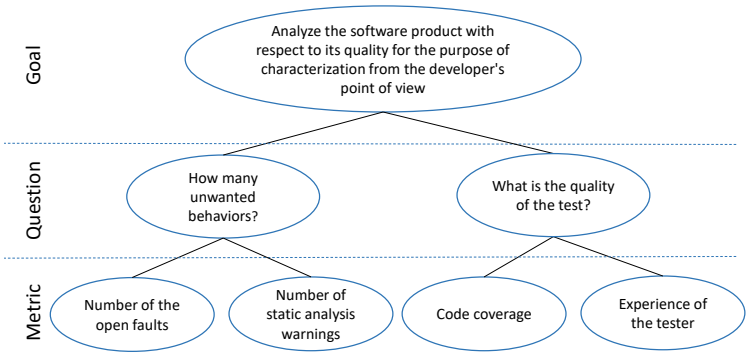


Figure 2. Example of a GQM model.

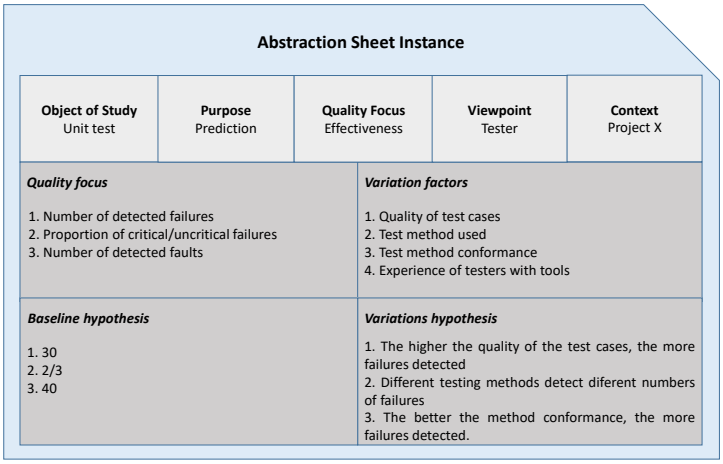


Figure 3. Abstraction sheet instance.

To execute this activity, we consider the validation criteria presented in Meneely et al. [14] due to its completeness. For each metric m related to the quality focus, the following steps must be performed:

1. Determine the intended use of the metric.
2. Highlight the advantages that are appropriate for the intended use chosen in the previous step. Some of these advantages are: mathematical soundness, practicality, correctness, efficiency and hypothesis-strengthening [14].
3. Look up the validation criteria that are tied to the advantages shown in Table 2.
4. Carefully choose validation criteria while considering the purpose of the metrics and the relationships and motivations among the criteria.
5. Analyze if the metric follows the chosen validation criteria.

#	Criterion	*Mathematical Soundness*	*Practicality*	*Correctness*	*Efficiency*	*Hypothesis-Strengthening*
1	*A Priori Validity*					X
2	*Actionability*		X			
3	*Appropriate Continuity*	X				
4	*Appropriate Granularity*					
5	*Association*					

Table 2. Example of mapping from criteria to advantages [14].

Given the concept of *variance* of ranked nodes, we can model the confidence in the validity of a set of metrics defined to answer a question. The greater the confidence that a set of metrics is valid to represent an attribute, the smaller that variance. To define the variance, the given rules of thumb should be used:

Rule 1: If the metric follows 100% of the validation criteria related to it, the variance must be equal to $5x10^-4$, the smallest value possible in AgenaRisk;

Rule 2: If the metric follows between 50% and 99% of the validation criteria related to it, the variance must be equal to $5x10^-3$;

Rule 3: If the metric follows between 1% and 49% of the validation criteria related to it, the variance must be equal to $5x10^-2$;

Rule 4: If the metric does not follow any validation criteria related to it, the variance must be equal to $5x10^-1$;

These intervals are recommendations for the first calibration of the Bayesian network and are restricted to AgenaRisk, which is currently the only software that supports ranked nodes. Another approach is to use the validation criteria as a reference and elicit knowledge from the domain expert to calibrate the variance. Furthermore, it is possible to, after applying the model, refine the calibration of the NPTs given knowledge from experts and collected data.

If a given node q has more than one parent node (i.e., more than one metric), the described rules should be applied considering the sum of the validation criteria for each metric.

The next step is to finalize the calibration of the NPT by, as presented in [8], defining a function to model the central tendency of the distribution that represents the NPT. To define the functions, weights and variance, knowledge must be elicited from the experts using the approach presented by Fenton et al. [8] or da Silva et al. [5].

4 Validation

We validated the resulting Bayesian networks in ten simulated scenarios. For all cases, we assumed that the first step of the proposed process was successfully executed. Due to space limitations, we only present the results of one scenario. This scenario describes a simple product quality model, where the goal is "Effectiveness of unit test', the question is "How many unwanted behaviors does the product have?' and the metrics are *number of detected failures* and *number of detected faults*.

In this case, given that the goal of using the metrics *number of opened faults* and *number of static analysis warnings* is to assist on decision-making during the development of the software, the advantage highlighted is *Decision-Informing*. There are 11 validation criteria associated with this advantage. On the other hand, say that the metric *number of opened faults* conforms to 8 out of 11 criteria and the metric *number of static analysis warnings* conforms to 5. Given this, together, both metrics conform to 13 out 22 validation criteria (i.e., 59.09%). Therefore, the second rule will be followed and the variance of the node *Unwanted behaviors* is $5x10^-3$.

To calibrate the NPT, we used the WMIN function, because if any of the given metrics are *Very low*, the answer to the corresponding question will tend to *Very low*. On the other hand, we considered *number of opened faults* as more important than *number of static analysis warnings*. Therefore, we defined them, respectively, with weights 2 and 1. We show an example of the calculated results for this scenario in Figure 4.

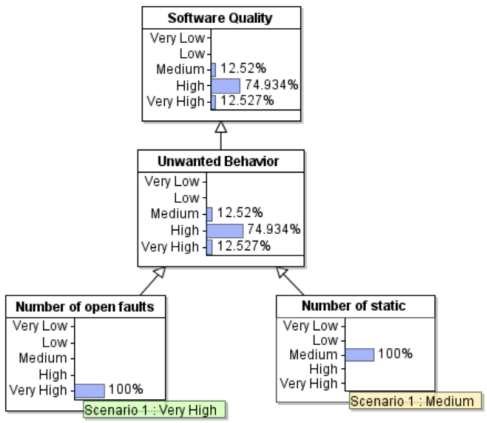


Figure 4. Example of a Bayesian network constructed using the proposed solution.

By analyzing Figure 4, it is possible to notice that, given the validity of the metrics used to answer the question, the confidence in the decision regarding the goal is acceptable. In case it was not, the probability of the goal would tend to be more uniform, meaning that a reliable decision could not be taken given the constructed model.

5 Final Remarks

In this paper, we presented a process to build Bayesian networks to represent the uncertainties of software metrics-based models. The process is composed of two activities: (i) metrics selection and (ii) metrics validation. The first activity is composed of three steps: characterization of the environment, acquisition of knowledge through abstraction sheets and construction of the Bayesian network. The second activity is composed of two steps: execution of the validation method from validation criteria and update the Bayesian network.

The process shown is based on the concept of ranked nodes [8] to build the Bayesian networks, with the goal of

adding meaning to metrics. Furthermore, it uses GQM to assist on the selection of metrics and software metrics validation criteria extracted from Meneely et al. [14]. On the other hand, if necessary, other types of nodes can be used such as Boolean, but it will be necessary to define a new reasoning to map the validity and the NPT definition.

The main limitation is the study's validation, which is only conceptual. On future works, we will execute empirical studies to evaluate our approach by collecting data from practitioners and tools to assess if the proposed solution improves the accuracy of decision making. Furthermore, we will complement our solution with additional steps regarding the definition of software metrics thresholds and collection reliability to assist on the construction of interpretation-oriented software metrics models.

References

[1] V. R. Basili. Software modeling and measurement: The goal/question/metric paradigm. Technical report, College Park, MD, USA, 1992.

[2] I. Ben-Gal. *Bayesian Networks*. John Wiley and Sons, 2007.

[3] L. C. Briand, C. M. Differding, and H. D. Rombach. Practical guidelines for measurement-based process improvement. *Software Process Improvement and Practice*, 2(4):253–280, 1996.

[4] A. Constantinou and N. Fenton. Towards smartdata: Improving predictive accuracy in long-term football team performance. *Knowledge-Based Systems*, pages –, 2017.

[5] R. da Silva, M. Perkusich, R. Saraiva, A. Freire, H. Almeida, and A. Perkusich. Improving the applicability of bayesian networks through production rules. In *27th International Conference on Software Engineering and Knowledge Engineering*, SEKE 2016, page In press, San Francisco, USA, 2016.

[6] C.-F. Fan and Y.-C. Yu. Bbn-based software project risk management. *Journal of Systems and Software*, 73(2):193–203, Oct. 2004.

[7] N. E. Fenton and M. Neil. Software metrics: roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 357–370. ACM, 2000.

[8] N. E. Fenton, M. Neil, and J. G. Caballero. Using ranked nodes to model qualitative judgments in bayesian networks. *IEEE Trans. on Knowl. and Data Eng.*, 19(10):1420–1432, Oct. 2007.

[9] L. Finkelstein and M. Leaning. A review of the fundamental concepts of measurement. *Measurement*, 2(1):25–34, 1984.

[10] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

[11] P. Hearty, N. Fenton, D. Marquez, and M. Neil. Predicting project velocity in xp using a learning dynamic bayesian network model. *Software Engineering, IEEE Transactions on*, 35(1):124–137, 2009.

[12] B. Kitchenham. Whats up with software metrics?–a preliminary mapping study. *Journal of systems and software*, 83(1):37–51, 2010.

[13] K. S. Mathias, J. H. Cross II, T. D. Hendrix, and L. A. Barowski. The role of software measures and metrics in studies of program comprehension. In *Proceedings of the 37th annual Southeast regional conference (CD-ROM)*, page 13. ACM, 1999.

[14] A. Meneely, B. Smith, and L. Williams. Validating software metrics: A spectrum of philosophies. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 21(4):24, 2012.

[15] M. Perkusich, A. Medeiros, K. C. Gorgônio, H. O. de Almeida, A. Perkusich, et al. A bayesian network approach to assist on the interpretation of software metrics. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 1498–1503. ACM, 2015.

[16] N. F. Schneidewind. Methodology for validating software metrics. *IEEE Transactions on software engineering*, 18(5):410–422, 1992.

[17] R. Shatnawi. Deriving metrics thresholds using log transformation. *Journal of Software: Evolution and Process*, 27(2):95–113, 2015.

[18] L. Uusitalo. Advantages and challenges of bayesian networks in environmental modelling. *Ecological Modelling*, 203(34):312 – 318, 2007.

[19] S. Wagner, A. Goeb, L. Heinemann, M. Kls, C. Lampasona, K. Lochmann, A. Mayr, R. Plsch, A. Seidl, J. Streit, and A. Trendowicz. Operationalised product quality models and assessment: The quamoco approach. *Information and Software Technology*, 62:101 – 123, 2015.

[20] L. G. Wallace and S. D. Sheetz. The adoption of software measures: A technology acceptance model (tam) perspective. *Information & Management*, 51(2):249–259, 2014.

Safe Incremental Design of UML Architectures

Anne-Lise Courbis[1], Thomas Lambolais[1], and Thanh-Hung Nguyen[2]

[1]LGI2P, IMT Mines Alès, France. firstName.lastName@mines-ales.fr
[2]Hanoi University of Science and Technology, Hanoi, Vietnam. hungnt@soict.hust.edu.vn

Abstract

IDF is an Incremental Development Framework which supports the development and the verification of UML models for reactive systems. IDF offers refinement and extension techniques allowing liveness properties to be preserved during the model developments. Here, we improve the framework in order to analyze models from a safety point of view. For this purpose, we associate IDF with the experienced tools of safety analysis based on the BIP language by translating UML models into BIP. We demonstrate on a basic example the complementarity of liveness and safety analyses.

Keywords*: UML composite components, BIP architectures, conformance analysis, safety analysis, refinement, incremental development.*

1. Introduction

Designing UML models of software intensive reactive systems is recognized to be a tricky and crucial task. Reactivity means that such systems must continuously react to their environment, at a speed defined by this environment. It implies *liveness properties*, stating that the system will eventually react as it must. These systems are also dependable, so that reliability, availability and robustness are of primary importance. This implies *safety properties*, stating that undesired behaviors of the system will never happen.

There is a lack of support for UML designers in the processes of both setting up models and evaluating them. The novelty of our proposed approach is to consider at early steps of the design both liveness and safety properties. In a previous work [8, 15, 14], we have presented our Incremental Development Framework (IDF) and its associated tool IDCM (Incremental Development of Conformance Models). IDF supports the development and the evaluation of UML models for reactive systems. It deals with UML composite components whose parts are composite or primitive components. Primitive component behaviors are described by UML state machines. IDF allows models to be developed step by step. At every step of the design, the model is verified as being consistent with the model obtained at the previous step. A step is a model evolution which can be of four kinds: *extension*, *refinement*, *increment* or *substitution*. The verification of these four kinds of model evolution is based on a conformance relation [16], which ensures that liveness properties of the former modeling step are preserved. However, this work has its own shortcomings: explicit verification of safety properties is not addressed. This article aims at enhancing IDF in order to be able to model and check explicit safety properties. This way, all temporal properties are considered, since the safety/liveness spectrum covers all linear temporal logic properties.

We present in section 2 an example pointing out an incremental development of a model whose liveness analysis is demonstrated using IDCM, but suffering from a lack of safety analysis. Section 3 presents the BIP intermediate format and the UML to BIP transformation in order to use D-Finder tool to analyze safety properties. The IDCM tool associated with IDF, as well as UML and BIP models presented in this article, may be downloaded on the website [7]. Section 4 presents related works. We conclude in section 5 and present our future directions.

2. Motivating example

Let us consider MUTEX, a mutual exclusion system that performs two task execution orders in parallel: it has two ports (Fig. 1), each of them allowing the reception of a task execution (start operation of IUserIn interface) and the transmission of an acknowledgement at the end (finish operation of IUserOut interface). The high level specification of MUTEX describes the system from an external point of

DOI: 10.18293/SEKE2017-033

view. The resource is not represented yet. It is modeled from a behavioral point of view by an atomic UML component (named SpecMUTEX) whose behavior is specified by a state machine with a concurrent state modeling two parallel task processes (Fig. 1).

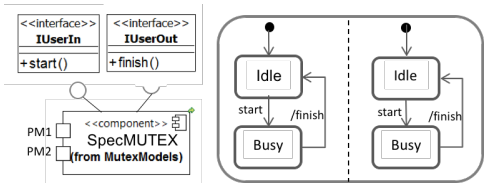
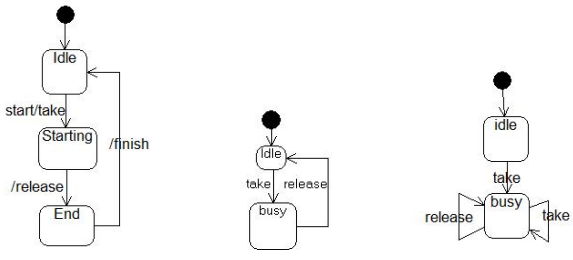


Figure 1: SpecMUTEX component

MUTEX1 model represents the first internal view of the system, i.e. two users sharing a resource. The resource provided interface has two operations: take and release. The designer thus specifies SpecUser and SpecResource components and their associated behaviors (Fig. 2a and b) in order to match SpecMUTEX specification. MUTEX1 is a composite component (Fig. 3a) which assembles two SpecUser components (U1 and U2) and one SpecResource component (R1) .

MUTEX2 model is a possible implementation model of MUTEX1 architecture. MUTEX2 has the same architecture than MUTEX1 (Fig. 3a) except that U1 and U2 are of type User and R1 is of type Resource (Fig. 2c). User details are not required to understand the example.

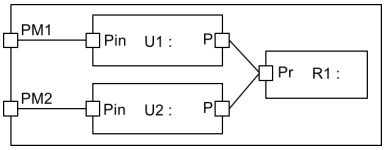
Both User and Resource are checked by IDCM as conforming their specification SpecUser and SpecResource (Fig. 3b). The conformance relation between an implementation model and a specification model guarantees that actions that are mandatory after any trace of the specification must also be accepted by the implementation after the same trace. This relation is implemented [16, 8] with its variant, refines and extends. It requires models to be transformed into LTS (Labelled Transition System). This is automatically achieved using IDCM [15, 14] which transforms the architecture into the EXP.OPEN formalism.



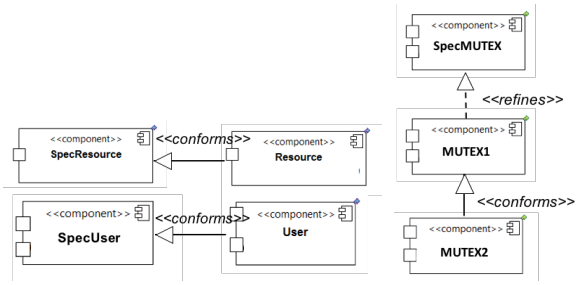
(a) SpecUser (b) SpecResource (c) Resource

Figure 2: MUTEX component state machines

Then, CADP [11] transforms EXP.OPEN into LTS. In the following, we give an interpretation of these relations on the MUTEX models (Fig. 3c). MUTEX2 component conforms to MUTEX1: it is a suitable implementation of the specification. MUTEX1 is a refinement of SpecMUTEX: it has no extra traces and must realize all mandatory behaviors of the specification.



(a) MUTEX1 and MUTEX2 architectures



(b) Implementation relations (c) Liveness relations

Figure 3: Incremental development of the MUTEX system

This example points out that MUTEX2 is a "good" realization of the initial specification from a liveness point of view, as defined by the ISO standard [12]. However, it does not verify the safety property stating that the resource has to be exclusive. This property states that U1 and U2 users must not at the same time be in the Starting state, which is the output state of the start/take transition, while R1 is in the Busy state .

It is necessary to associate another tool with IDCM to verify such a property.

3. Safety analysis of architectures

Our goal is to explicitly specify safety properties in order to complement incremental techniques. We are looking for a method which takes into account the incremental aspect of modeling in order to be integrated into IDF. The BIP (Behavior, Interaction and Priority) modeling and verification framework [2] is based on several principles which match with IDF: development of correctness-by-construction model, incrementality, compositionality and composability. It includes many tools for safety property analysis and model transformation from several languages such as AADL and Lustre. To the best of our knowledge, there is no available transformation tool from UML to BIP taking into account both primitive and composite compo-

nent descriptions. We have thus developed a module of IDCM that may be uploaded on the IDCM website [7]. We present an overview of the transformation from UML to BIP and the benefits for safety analysis of architectures.

3.1. From UML to BIP

BIP [1] is a powerful language for modeling heterogeneous real-time systems. Main classes of BIP meta-model corresponding to UML concepts we deal with are: Compound Type, Atomic Type, State, Transition, Port and Connector. BIP describes compound components by a set of interactions between atomic components whose behaviors are represented by LTS. There is thus a direct mapping between BIP and UML atomic components since we gave in [14] a LTS semantics to UML models. Listing 1 gives the corresponding BIP model automatically generated by the IDCM module UMLtoBIP.

```
package Resource
atomic type Resource
  export port Port PR_TAKE
  export port Port PR_RELEASE
  port Port i
  place Pseudostate1, idle, busy
  initial to Pseudostate1
  on i from Pseudostate1 to idle
  on PR_TAKE from idle to busy
  on PR_TAKE from busy to busy
  on PR_RELEASE from busy to busy
end
end
```

Listing 1: Resource BIP model

There is a direct mapping between UML composite component and BIP compound components: a UML composite component consists of a set of *Parts* which match BIP *Components*. A UML assembly *Connector* matches a set of *BIP connectors*. Indeed, a BIP connector is relative to the synchronization of a single operation shared by two interconnected ports, while a UML connector is relative to the synchronization of the set of operations belonging to interfaces associated with the interconnected ports. The *Port* of a *Part* belonging to a delegate connector will be exported and renamed by the name of the port of the compound component. To illustrate this transformation, we give in Listing 2 the BIP code of the MUTEX2 architecture presented in section 2. This code is automatically generated by the transformation IDCM module UMLtoBIP.

3.2. D-Finder: a toolbox for safety analysis

D-FINDER provides methods and tools to compute invariants of BIP models. Such invariants are interesting since they preserve safety properties. There are two kinds of invariants: component invariants which are over approximations of reachable states, and interaction invariants which

```
model Mutex
include User.bip
include Resource.bip
connector type RDV(Port p1, Port p2)
        define [p1 p2] end
compound type MutexType
  component Resource R1
  component User U2 component User U2
  connector RDV C1_release(U1.P_RELEASE, R1.PR_RELEASE)
  connector RDV C1_take(U1.P_TAKE, R1.PR_TAKE)
  connector RDV C2_release(U2.P_RELEASE, R1.PR_RELEASE)
  connector RDV C2_take(U2.P_TAKE, R1.PR_TAKE)
  export port Port PM1_FINISH is U1.PIN_FINISH
  export port Port PM1_START is U1.PIN_START
  export port Port PM2_FINISH is U2.PIN_FINISH
  export port Port PM2_START is U2.PIN_START
end
component MutexType Mutex
end
```

Listing 2: Mutex BIP model

define global boolean constraints dealing with the synchronization of components. D-FINDER is based on an abstraction technique allowing the state space to be reduced. Its strength is to perform incremental constructions of models and incremental computations of invariants [2] allowing large-scale systems to be checked. D-Finder uses the BDD library for the symbolic computation of interaction invariants, and then the SAT-solver tool Yices [9] for checking satisfiability. Verifying a safety property consists in demonstrating using Yices that the negation of the property is unsatisfiable in a context defined by the set of invariant expressions generated by D-Finder. The invariants are expressed by Boolean Behavioral Constraints [17].

3.3. Illustration of a safety property for MUTEX

Let us consider MUTEX1 and MUTEX2 architectures presented in section 2. We aim at checking the mutual exclusion property. Equation (1) expresses the non expected property: two users U1 and U2 can be both in Starting state while the shared resource R1 is in busy state.

$$(and\ \text{R1\_busy-}\ (and\ \text{U1\_Starting-}\ \text{U2\_Starting-})) \quad (1)$$

For MUTEX1, the property is unsatisfied: it means that the resource mutual exclusion has been properly implemented in this architecture. That is not the case for the MUTEX2 architecture. Indeed, the Resource state machine (Fig.2c) points out that it may be used concurrently by two users. On this example, the error is obvious, but it is not the case for large systems where components may be designed by third parties according to a high specification level.

4. Discussion and Related work

To the best of our knowledge, no framework support the *incremental* development of UML architecture mod-

els by analyzing both *liveness and safety* behavioral aspects. In particular, no framework is able to consider abstract and non-deterministic UML models and few frameworks are able to consider UML models partially covering the requirements. Hence, even if some work addresses refinement of models, they do not focus on the reduction of non-determinism, and most of them cannot analyze models the specification of which is extended, despite it is a key action for designing complex systems and managing model evolution. For example, [13] defines a UML profile to transform models into Wright for using the FDR model checker [10].[18] proposes to translate architectures into IF/IFx models [5] allowing LTS models to be generated and analyzed by the CADP model checker [11].Other approaches about AADL aims at transforming models into intermediate models such as FIACRE [6, 3] or BIP [6] to use appropriate model checkers such as TINA [4] or Yices [9].

Refer to [14] to have more arguments and a complete state of the art about formal verification of models.

These approaches are powerful from the safety point of view but they are not able to integrate liveness analysis for incremental development of models as it is done in IDCM.

5. Conclusion

In this article, we have pointed out the interest for incremental development of UML models and the complementarity between safety and liveness analyses. We have defined a transformation of UML models into the BIP formalism which is implemented into the tool IDCM we have developed. By this way, the liveness and safety analyses are automated. Safety properties are expressed by propositional calculus and requires designers to manipulate a specific syntax. Further steps consist in developing a support to help designers to express the safety properties in terms of UML concepts regardless the theorem prover syntax, and studying their automatic rewording when UML models are refined or extended.

Acknowledgement: This research was supported by IMT Mines Alès–France through its mobility funding program and the National Foundation for Science and Technology Development (NAFOSTED) under Grant 102.03-2013.39: Automated verification and error localization methods for component-based software.

References

[1] A. Basu, M. Bozga, and J. Sifakis. Modeling heterogeneous real-time components in BIP. In *SEFM*, pages 3–12, 2006.

[2] S. Bensalem, M. Bozga, A. Legay, T.-H. Nguyen, J. Sifakis, and R. Yan. Incremental component-based construction and verification using invariants. In *FMCAD*, pages 257–266, 2010.

[3] B. Berthomieu and J.-P. Bodeveix. Formal Verification of AADL models with Fiacre and Tina. In *ERTS*, 2010.

[4] B. Berthomieu, P.-O. Ribet, and F. Vernadat. The tool TINA: Construction of abstract state spaces for Petri nets and time Petri nets. *International Journal of Production Research*, (14):2741–2756, 2004.

[5] M. Bozga, S. Graf, I. Ober, I. Ober, and J. Sifakis. The IF Toolset. In *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *LNCS*, pages 237–267. Springer Berlin Heidelberg, 2004.

[6] M. Y. Chkouri and M. Bozga. Prototyping of distributed embedded systems using AADL. *ACESMB*, pages 65–79, 2009.

[7] A.-L. Courbis, T. Lambolais, H.-V. Luong, and T.-L. Phan. IDCM. `http://idcm.wp.mines-telecom.fr`. Accessed: 2017-05-05.

[8] A.-L. Courbis, T. Lambolais, H.-V. Luong, T.-L. Phan, C. Urtado, and S. Vauttier. A formal support for incremental behavior specification in agile development. In *The 24th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pages 694–699, 2012.

[9] B. Dutertre. Yices 2.2. In A. Biere and R. Bloem, editors, *CAV*, volume 8559 of *LNCS*, pages 737–744. Springer, July 2014.

[10] Formal-Systems and Oxford-University-Computing-Laboratory. Failures-Divergence Refinement (FDR2 User Manual). Technical Report October, Formal System (Europe) Ltd, 2010.

[11] H. Garavel, F. Lang, R. Mateescu, and W. Serwe. CADP 2011: a toolbox for the construction and analysis of distributed processes. *International Journal on Software Tools for Technology Transfer*, 15(2):89–107, 2013.

[12] ISO/IEC9646. Information technology – open systems interconnection – conformance testing methodology and framework – part 1: General concepts, 1991.

[13] M. Kmimech, M. T. Bhiri, and P. Aniorte. Checking component assembly in ACME: an approach applied on UML 2.0 components model. In *ICSEA*, pages 494–499. IEEE, 2009.

[14] T. Lambolais, A.-L. Courbis, H.-V. Luong, and C. Percebois. IDF: A framework for the incremental development and conformance verification of UML active primitive components. *Journal of Systems and Software*, 113:275–295, 2016.

[15] T. Lambolais, A.-L. Courbis, H.-V. Luong, and T.-L. Phan. Designing and integrating complex systems: Be agile through liveness verification and abstraction. In *CSDM*, pages 69–81. Springer, 2015.

[16] H.-V. Luong, T. Lambolais, and A.-L. Courbis. Implementation of the Conformance Relation for Incremental Development of Behavioural Models. In *MoDELS*, volume 5301 of *LNCS*, pages 356–370. Springer Berlin, 2008.

[17] T.-H. Nguyen. *Constructive verification for component-based systems*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2010.

[18] I. Ober and I. Dragomir. Unambiguous UML composite structures: the OMEGA2 experience. In *SOFSEM*, pages 418–430. Springer, 2011.

Enhancing Sample-based Scheduler with Collaborate-state in Big Data Cluster

Chunliang Hao [✦✧] Celia Chen [★] Jie Shen [☆] Mingshu Li [✦] Barry Boehm [★]

[✦]Institute of Software, Chinese Academy of Science, Beijing, China
[★]Center for Systems and Software Engineering, University of Southern California, USA
[☆]Department of Computing, Imperial College London, UK
[✧]University of Chinese Academy of Science, China
Email: chunliang@nfs.iscas.ac.cn, qianqiac@usc.edu, js1907@imperial.ac.uk,
mingshu@iscas.ac.cn, boehm@usc.edu

Abstract

Sample-based scheduler design has become an emerging research topic for its high scalability and simple scheduling process in today's big data cluster. One major limitation of such design is its lack of global cluster knowledge, which leads to sub-optimal decisions. Some cutting edge schedulers solve this issue by deploying an extra centralized component in the cluster to capture the real-time cluster state and inform all schedulers. However, such solution is with high cost and low scalability. As an alternative, we introduce the Collaborated-Cluster State(CCS) technique in this paper. CCS is a low cost solution that merely harms the scalability of sample-based design, while achieving similar performance gain as ECC. Experiments with Google and Yahoo production trace both show that CCS under most scenarios can keep up with ECC's performance while reducing 87.7% (in Google trace) and 73.9% (in Yahoo trace) of communications.

1 Background and Introduction

Sample-based scheduling is currently one of the most promising branches of distributed scheduling. It is both fully distributed and low-latency. Currently, the cutting edge sample-based schedulers adopt the batch-probing sampling process proposed by Sparrow. The batch-probing sampling process contains following key steps: 1. In order to schedule a t-task-job, a scheduler randomly samples $2t$ nodes from the cluster (the ratio of sampling of 2 follows the power of two law[1], but could also be changed to other value). 2. The scheduler sends each selected node a probe request. 3. Each probe is then queued in the worker node to serve as task reservation. 4. When a task reservation is at the head of the queue and is ready to execute, the worker replies to the scheduler and gets a task to run. 5. Once the scheduler sends all t task to run, it cancels all remaining reservations for the job by notifying according workers. 6. Worker will notify the scheduler of the completion of the assigned task. More details could be found in the Sparrow paper[2].

One crucial problem of these sample-based schedulers is their lack of global knowledge of the cluster status. During each decision process, a scheduler can only communicate with a very small number (e.g. $2t$ in above example) of sampled worker nodes and makes decision based on information gathered from those nodes.This limits the sample-based scheduler and leads to sub-optimal task placement decision.

An example of sub-optimal decisions made by sample-based schedulers is shown in Figure 1. In this example, the scheduler has randomly sampled two busy worker nodes, hence the task under schedule will experience queue waiting no matter which worker is chosen. However, there are six available worker nodes elsewhere in the cluster, this queue waiting could have been avoided if the scheduler had that information. Such a decision is considered sub-optimal since there are better decisions that existed and should be found in the cluster. However, for most cases scheduler itself can't realize a sub-optimal decision is been made; for instance scheduler in both Figure 2 and Figure 1 will consider themselves in the same status.

Some cutting edge scheduler mitigate this problem by using a centralized software component to synchronize cluster status and push collected global cluster knowledge to each sample-based scheduler (refer to as EXCC).

This centralized component can either be a share-state master, which is specifically designed to synchronize task and resource status with all worker nodes, or an independent centralized scheduler, which is responsible to schedule and inform the sample-based schedulers at the same time. Either way, the centralized component has to synchronize with all worker nodes to capture the real-time status of the cluster and then inform all the sample-based schedulers accordingly.

Although this EXCC solution have proven to effectively reduce sub-optimal decision and improve scheduling precision, EXCC is very expensive to implement. Since the centralized component is required to synchronize with all workers in cluster in order to collect global cluster information, the component itself becomes a potential system bottleneck, which is the pri-

DOI reference number: 10.18293/SEKE2017-136

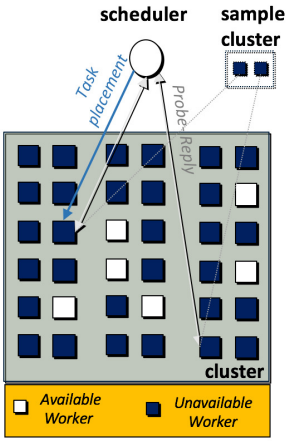


Figure 1: A sub-optimal task placement decision.

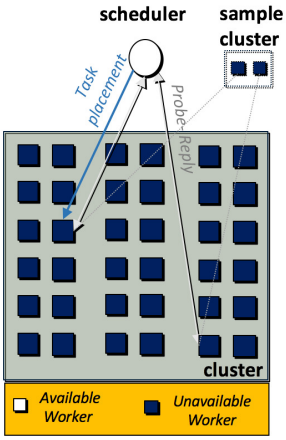


Figure 2: A good task placement decision.

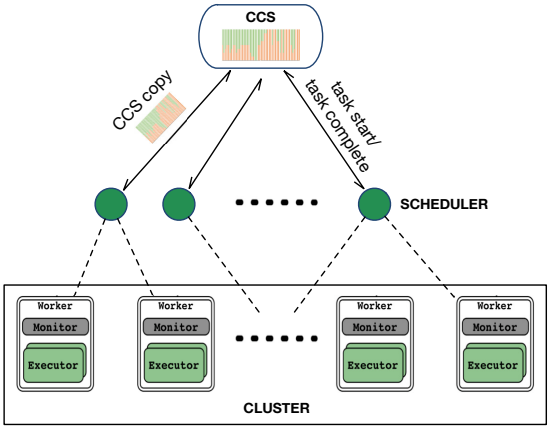


Figure 3: Overview of sample based scheduling with CCS

mary goal for sample-based schedulers to avoid. Moreover, a large amount of extra communications is needed to capture the real-time status of cluster.

In this paper, we propose collaborated-cluster-state (CCS), an alternative design that is low-cost and scalable, which allows the scheduler acquire global knowledge while keeping the simplicity of sample-based scheduling design.

2 Collaborated-Cluster-State

In this paper, we introduce a novel technique, collaborated-cluster-state (CCS), which provides sample-based schedulers with global cluster knowledge through scheduler collaboration. CCS keeps track of all occupied and available resource status by receiving updates from schedulers only. Each scheduler packs task start and complete information along with necessary task details together and send it to CCS periodically. CCS also periodically pushes its current accumulated knowledge to all schedulers. This pushed copy of CCS knowledge can provide each scheduler with advice on which workers should be probed and which should not.

In CCS, cluster resource information is stored in the form of an array of size N, in which N represents the number of worker nodes in the cluster. Each element in the array caches the resource status of the corresponding worker node in the cluster in a tuple $(\vec{ra}, \vec{ro})$, where $\vec{ra}$ denotes the available resource quantity in a worker node; $\vec{ro}$ denotes the occupied resource quantity in a worker node. For instance, $CCS[10] = (3, 4, 500, 5, 12, 200)$ means that the 10^{th} worker node in the cluster has 3 core, 4GB of DRAM and 500GB of Flash available and 5 core, 12GB of DRAM and 200GB of Flash occupied.

The overall design to maintain and use CCS knowledge in sample-based scheduling is shown in Figure 3.The CCS component needs to be initialized before each scheduler starts to function. Upon initialization, the CCS component reads in worker registration information from the cluster daemon to acquire the maximum resource capacity of each worker node and marks them as available.

After initialization, CCS component first waits for incoming messages from each scheduler. When a message arrives, CCS unpacks the message to recover the task start or complete information. Each piece of information is represented as one tuple. CCS then processes these tuples one by one. For each tuple, the component reads in the following information: whether the tuple represents task start or complete; the worker nodes of which each task runs on; the amount of resources each task claims. CCS then finds that worker node in its data and updates the amount of available and occupied resources accordingly. CCS also pushes a copy of the cluster state knowledge to all schedulers periodically. The interval of push, ω, is preset, where smaller ω value makes copies in each scheduler more precise but requires more communication while bigger ω value leads to the opposite situation. Each CCS copy also has an expire period β, to prevent the failure of CCS component.

Each scheduler receives a CCS copy and only keeps the latest version. At the beginning of each scheduling process, instead of choosing sample target at complete random, it checks with its own CCS copy first. Suppose $2t$ workers need to be selected for a t-task-job and each task claims resources $\vec{rc}$, the scheduler finds in its CCS copy for $2t$ worker that has $CCS[worker].\vec{ra} \geq \vec{rc}$. If more than $2t$ workers are found to be qualified, the scheduler chooses workers with most available resource by default. If less than $2t$ qualified workers are found, the scheduler chooses from the rest of workers randomly.

After sample target is chosen, the rest of decisioning process follows the common batch-probing process. The scheduler probes towards selected workers and places task reservations. When the scheduler is notified by any worker that one reservation is ready to execute, the scheduler places a task with that worker. It keeps doing so until all t tasks are placed. Once all tasks are placed, it cancels all the leftover reservations in the cluster. When the scheduler places task towards a worker, it caches a tuple $[1, workerID, rc]$ locally. When the sched-

Table 1: Number/distribution of job and task in trace

	Google(2011)	Yahoo(2011)
Number of Jobs	506.4k	24.2k
Number of Tasks	17889.7k	968.3k
% Jobs ed $\leq$ 1000s	89.0%	96.6%
% Jobs ed $\leq$ 100s	42.7%	36.4%
% Jobs ed $\leq$ 10s	0.0%	2.7%
% Tasks ed $\leq$ 1000s	69.7%	84.6%
% Tasks ed $\leq$ 100s	7.6%	54.0%
% Tasks ed $\leq$ 10s	0.0%	20.2%

uler is notified that a task is completed, the scheduler caches a tuple $[0, workerID, rc]$ locally. The first variable in tuple represents task start/complete, where 1 represents start and 0 represents finished. The second variable is the ID of the current worker, and the last variable is the amount of resources the task claims. Periodically (with an fixed interval γ) it packs all cached tuples together and sends them to CCS. Each tuple will be sent only once.

3 Experiments

3.1 Methodology

Workload: Google trace and Yahoo trace are used in the evaluation as a representation of today's production workloads in big data cluster. The Google trace used in this paper[3][4] is publicly available. Cleaning the trace by removing failed or invalid records resulted in more than 500k heterogeneous jobs. Task runtime also varies within each job. For Yahoo trace, we gathered centroid values for task duration and average number of task per job from the trace description[5][6], then used them as scale parameters in exponential distribution to generate the complete trace. The job/task distribute of Google trace and Yahoo trace are listed in Table 1.

Metrics: In this paper, we used three metrics to evaluate CCS against some baseline techniques. We first evaluated how many sub-optimal decisions a sample-based scheduler could avoid with CCS and then collated the job runtime results of CCS and of the baseline techniques in Google and Yahoo trace. 50^{th} and 90^{th} percentile job runtime of the workload were used in this experiment. Finally we compared the communication costs of CCS against the baseline schedulers.

Baseline Schedulers and Simulator: There are two baseline schedulers used in this study: Sparrow and EXCC. Sparrow was used to represent the cutting edge sample-based scheduler without any global cluster knowledge. In the experiment, we use the event-based Sparrow simulator from its own open-source project[2] and further augmented this event-based scheduler for the evaluation of CCS. It was used as the lower bound in performance and communication cost. EXCC represented the high-cost solution that uses a centralized component to obtain a more precise global cluster state for the sample-based schedulers. It was used to represent the upper bound in sub-optimal decision and lower bound in job run-time. We abstracted the EXCC used in Tarcil[7] scheduler for the evaluation of CCS.

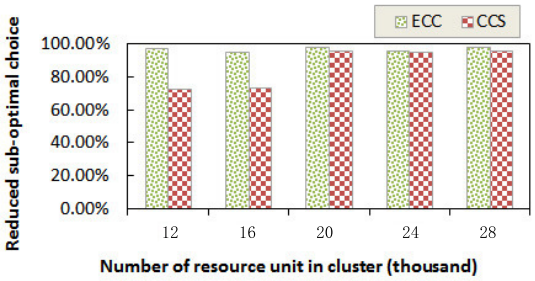


Figure 4: The percentage of reduced sub-optimal decision using CCS and EXCC, comparing to Sparrow, Google trace

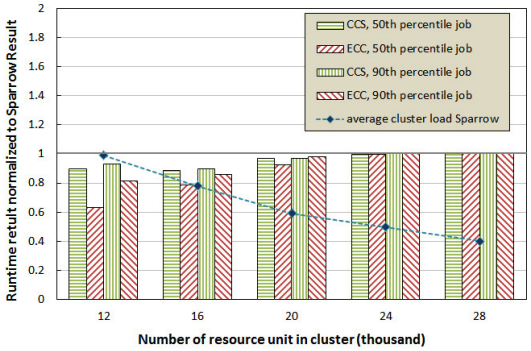


Figure 5: Google trace runtime results in five different cluster; evaluating Sparrow, CCS and EXCC. Both CCS and EXCC results are normalized to Sparrow.

Cluster setting: We changed the size of the cluster to achieve different cluster utilizations and simulate the industrial workloads. The number of schedulers deployed in the cluster was set to 10. For CCS, the parameter ω was set to 10 second and γ was set to 5 second. Probe ratio was fixed at 2 following the Power of Two Law[1].

3.2 Google Trace Results

The sub-optimal decision statistics from Google Trace are shown in Figure 4. Both CCS and EXCC effectively reduced most of the sub-optimal decisions in sample-based scheduling. Especially when the cluster was under medium and low loaded situation (20k,24k,28k slots cluster), almost all of the sub-optimal decisions were avoided. In high cluster, CCS made sub-optimal decisions mostly because of its imprecise global knowledge; EXCC because of conflicts.

The result of job runtime of Google trace is shown in Figure 5. We normalized the runtime results of CCS and EXCC to Sparrow for better comparison. In this case, smaller results indicated better performance improvement. As shown in the figure, the major improvement of both CCS and EXCC was in high and extreme loaded (16k and 12k slots) clusters. The difference between EXCC and CCS was small except in extreme loaded cluster, where EXCC outperformed CCS by about 20%.

The overall communication count in/out of CCS component

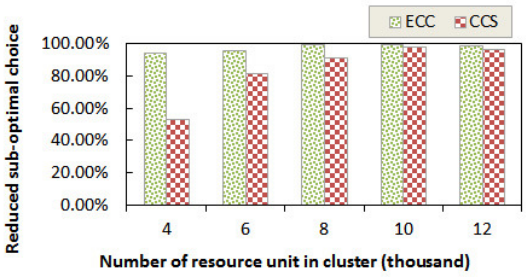


Figure 6: The percentage of reduced sub-optimal decision using CCS and EXCC, comparing to Sparrow, Yahoo trace

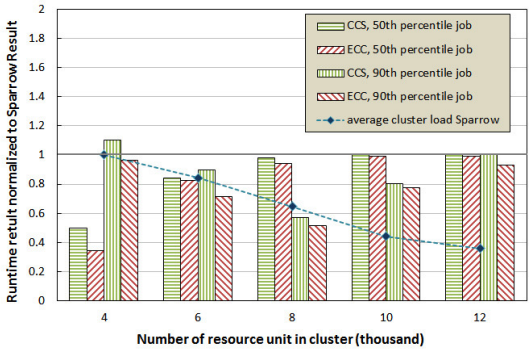


Figure 7: Yahoo trace runtime results in five different cluster; evaluating Sparrow, CCS and EXCC. Both CCS and EXCC results are normalized to Sparrow.

in above experiments was roughly 7.5 million in total, while the EXCC required about 60.6 million in total. When the throughput of workload and cluster size increases (as expected in future's big data cluster), the communication needed by EXCC increases accordingly while CCS stays the same.

3.3 Yahoo Trace Results

Yahoo trace results showed very similar trends as Google results. Compared to Sparrow, both CCS and EXCC reduced almost all sub-optimal decisions. CCS had a very similar performance as EXCC, as shown in figure 6.

Considering runtime results, the results showed that both CCS and EXCC reduced about 20% job runtime of Sparrow in high loaded cluster(6k slots) and cut 90th percentile runtime of Sparrow in half in medium loaded cluster(8k slots).

Similar to Google trace results, CCS needed much less communication than EXCC. The total amount of communication in/out of CCS is about 0.55 million and EXCC is about 2.11 million.

The results from both Google and Yahoo trace show that CCS achieved similar performance using far less (respectively 86.7% and 73.9% less) communication cost. The impact of latency (a 10 second pushing delay and a 5 second synchronization delay) in CCS has a limited impact on the final runtime result, as we expected.

In above experiments, while EXCC had to synchronize with thousands of work nodes, CCS only needed to synchronize with 10 worker nodes. Compared to EXCC, the packing of updates in CCS changed synchronization frequency from per-task frequency to a fixed rate, which in each experiment was from about 1 per second in EXCC to 0.2 per second in CCS.

4 Related Work

Among existing sample-based methods, Sparrow is one of the most popular and represented work with the batch-probing technique. In this paper, we have introduced how to implement CCS through augmenting Sparrow and also used Sparrow scheduler as the baseline in our experiment. Tarcil is also a distributed, sample-based scheduling approach. It uses extra centralized components to combine the benefit of share-state design and sample-based design. We have compared the performance of such EXCC method and our proposed CCS method.

Another direction of providing sample based scheduler with global knowledge is the hybrid design. Eagle[8] proposed a design that uses a EXCC as a scheduler for long jobs and also as a source of global information, which notifies sample based schedulers about the placement of long tasks in cluster.

5 Conclusions

Collaborated-cluster State (CCS) is a novel technique to improve the precision of sample-based schedulers by providing them with global cluster knowledge. Comparing to existing techniques that serve the same purpose, it requires much lower communication cost while preserving the scalability of sample-based design.

Acknowledgement

This work is financially supported by the Strategic Priority Research Program of the Chinese Academy of Science (No. XDA06010600), as part of the DataOS project.

References

[1] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE TPDS*, vol. 12, no. 10, pp. 1094–1104, 2001.

[2] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow: distributed, low latency scheduling," in *SOSP*, 2013, pp. 69–84.

[3] J. Wilkes. More google cluster data. [Online]. Available: http://googleresearch.blogspot.ch/2011/11/more-google-cluster-data.html

[4] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *SoCC*, 2012, pp. 7–13.

[5] Y. Chen, S. Alspaugh, and R. Katz, "Interactive analytical processing in big data systems," in *VLDB*, 2012, pp. 1802–1813.

[6] Y. Chen, A. Ganapathi, R. Griffith, and R. H. Katz, "The case for evaluating mapreduce performance using workload suites," in *MASCOTS*, 2011, pp. 390–399.

[7] C. Delimitrou, D. Sanchez, and C. Kozyrakis, "Tarcil: reconciling scheduling speed and quality in large shared clusters," in *SoCC*, 2015, pp. 97–110.

[8] P. Delgado, D. Didona, F. Dinu, and W. Zwaenepoel, "Job-aware scheduling in eagle: Divide and stick to your probes," in *SoCC*, 2016.

A Stratification and Sampling Model for Bellwether Moving Window

Solomon Mensah[1], Jacky Keung[1], Michael Bosu[2], Kwabena Bennin[1] and Patrick Kwaku Kudjo[3]
[1]Department of Computer Science, City University of Hong Kong, Hong Kong, China
[2]Centre for Business, Information Technology and Enterprise, Wintec, Hamilton, New Zealand
[3]Department of Computer Science and Communication Engineering, Jiangsu University, China
{smensah2-c, kebennin2-c}@my.cityu.edu.hk, Jacky.Keung@cityu.edu.hk,
michael.bosu@wintec.ac.nz, kudjo@ujs.edu.cn

***Abstract*—An effective method for finding the relevant number (window size) and the elapsed time (window age) of recently completed projects has proven elusive in software effort estimation. Although these two parameters significantly affect the prediction accuracy, there is no effective method to stratify and sample chronological projects to improve prediction performance of software effort estimation models. Exemplary projects (*Bellwether*) representing the training set have been empirically validated to improve the prediction accuracy in the domain of software defect prediction. However, the concept of *Bellwether* and its effect have not been empirically proven in software effort estimation as a method of selecting exemplary/relevant projects with defined window size and age. In view of this, we introduce a novel method for selecting relevant and recently completed projects referred to as *Bellwether moving window* for improving the software effort prediction accuracy. We first sort and cluster a pool of *N* projects and apply statistical stratification based on Markov chain modeling to select the *Bellwether moving window*. We evaluate the proposed approach using the baseline Automatically Transformed Linear Model on the ISBSG dataset. Results show that (1) *Bellwether effect* exist in software effort estimation dataset, (2) the *Bellwether moving window* with a window size of 82 to 84 projects and window age of 1.5 to 2 years resulted in an improved prediction accuracy than the traditional approach.**

Keywords—Bellwether effect; Markov chains; Window age; Window size; Chronological dataset.

1. INTRODUCTION

In the construction of a robust and accurate prediction model, there is the need to consider the best selection of training and validation sets prior to modeling. According to Lokan and Mendes [5][12], selection of training and validation sets from a subset of chronologically arranged projects (referred to as *moving window*) can further improve the prediction accuracy of effort estimation models. The moving window theory is based on the assumption that recent projects are likely to share similar characteristics with *new* projects. Recent studies [10][2][7][17] support the theory by Lokan and Mendes that, *the use of moving window as the training set improves the estimation accuracy of effort estimation models*. The moving window approach has recently been supplemented with *weighting functions which further improve the estimation accuracy of effort estimation models.* This theory was postulated by Amasaki and Lokan [10][2][18][17] and was empirically confirmed to improve the estimation accuracy when using large-sized moving windows. These previous studies found software effort estimation (SEE) models built using recently completed projects (weighted/unweighted moving window) superior to models that use all available historical projects (referred to as the *growing portfolio*).

Irrespective of the significant improvement of using weighted/unweighted moving windows, challenges still exist in determining how large and how old the moving window should be prior to SEE modeling. The *window size* simply refers to the number of most recently completed historical projects appropriate for building the SEE model. The *window age* refers to the elapsed time of projects (within the moving window) that has existed for not more than *t* calendar years or calendar months [7]. Previous studies [2][5][10][18] randomly used different moving window sizes such as 20, 30, 40…120 projects. Similarly, different moving window ages (such as 1 year, 1.5 years, 2 years…) were used by previous studies [7][12][17]. These arbitrary window sizing and window aging parameters significantly affect the estimation accuracy irrespective of the weighting functions used [2][10]. This is mainly due to the variations in the sample sizes drawn from the chronological datasets. Based on previous moving window studies [10][5][2][12], this study aims at addressing the unstable window sizing and window aging constraint to further improve the estimation accuracy of predictive models. The existence of exemplary projects referred to as *Bellwether* has successfully been used to improve the relative estimation accuracy in software defect prediction by Krishna et al. [1].

We therefore seek to address this problem by introducing a novel model to sort, stratify and statistically sample exemplary and recently completed historical projects to be considered as the *Bellwether moving window*. The model operates by applying the concept of *Bellwether* [1] and a probabilistic modeling approach namely Markov chain Monte Carlo [6] to determine an effective size and age of the moving window used in modeling.

Based on Markov chain modeling, the outcome of an event (*new* project) in an experiment depends only on the previous experiment (recently completed projects with defined transition probabilities). The concept of *Bellwether* (specifically the *Bellwether effect*) and the Markov chains theory [6] are empirically proven to determine the best window age and window size of moving windows in this study.

In this study, we make the following contributions: To the best of our knowledge, this is the first study to 1) empirically prove the existence of *Bellwether* in software effort estimation, and 2) compute the best window size and window age of the

DOI reference number: 10.18293/SEKE2017-126

sampled *Bellwether moving window* based on Markov chain Monte Carlo.

This study is unique since the *Bellwether method* (as elaborated in Section 4) addresses the window aging and sizing constraint by developing a step-by-step sampling method as well as computing the window size and window age of the *Bellwether moving window* from a set of chronological projects.

The remaining sections of the paper are organized as follows. Section 2 presents the *Bellwether* concept and the Markov chain Monte Carlo approach. Section 3 presents the postulations with their respective proofs. The proposed approach is presented in Section 4. Section 5 details the methodological procedure. Section 6 presents the experimental results. Section 7 presents the threat to validity and Section 8 concludes the study.

2. BACKGROUND

A. The Concept of Bellwether

According to Krishna et al. [1], the concept of *Bellwether* which is a simple transfer learning technique is defined in 2-folds namely the *Bellwether effect* and the *Bellwether method*:

1) The Bellwether effect states that, given a set of N projects, there exist exemplary project(s) that can form the Bellwether and can provide the best prediction accuracy for the remaining N-1 projects.

2) The Bellwether method uses a heuristic approach to search for that particular Bellwether from a pool of completed projects and applies it to a new project whose target is to be estimated.

This concept of *Bellwether* was considered by Krishna et al. [1] in the domain of defect prediction whereby given a set of non-chronological projects, each project was used as a potential *Bellwether* to successfully make predictions on the remaining projects. Results from their study show that *Bellwethers* could obtain the best training sets with relative improved prediction accuracy for building estimation models.

This paper therefore investigates the feasibility of using *Bellwether* as a *moving window* for building software effort estimation models. We first, sort the projects chronologically and apply a statistical stratification technique to obtain only the recently completed projects. We then check for the existence of *Bellwether* based on a defined *Bellwether method* as elaborated in Section 4. We define the moving window obtained from the Bellwether method as the *Bellwether moving window.* The *Bellwether moving window*, w_i is used to estimate each of the remaining windows, w_j in $N \; \forall \; i \neq j$.

B. Markov chain Monte Carlo

The Markov chain Monte Carlo (MCMC) is a probabilistic technique that seeks to solve the problem of sampling by exploring a given sample space through the construction of an ergodic Markov chain (EMC) whose limiting distribution is the target distribution [6]. MCMC utilizes Markov chains to effectively simulate a random variable *X* whose future states are independent of past states given the present state. MCMC has been applied and proven successful in different software engineering domains such as software testing [8], image processing [11] as well as applied mathematics, statistical and biomedical engineering.

We provide a statistical investigation on the use of MCMC to obtain a potential subset of relevant projects to be considered as the *Bellwether*. Let *D* denotes a set of projects from a given population (industry) with *d* degree of dimensions (features). Then, $X = (x_1, \ldots, x_d) \epsilon D$ can represent a sample whose limiting distribution is known and can form the *Bellwether*. Each of the sample points $x_1, \ldots, x_d$ of the sample is allowed to take a discrete or continuous time value which denotes the project ages already known from the given repository *(D)*. The Markov chain is then executed a number of times until the chain converges to its limiting distribution to obtain the target sample subset [6]. The selected sample with its respective *t* states (or ages) and constructed limiting distribution forms the *Bellwether* which can be used as the moving window. We define this type of moving window as the *Bellwether moving window.*

3. POSTULATIONS

In this study, we prove the following three postulations based on empirical analysis that:

1) Given a set of N chronological projects with a finite mean μ and variance σ^2, then there exist a potential Bellwether which can be used as a moving window.

Proof: Let $\{X^{(1)}, \ldots, X^{(t)}\}$ be independent and identically distributed (*iid*) random samples chronologically drawn from a given population, *N*. If a random sample $X^{(i)} > 30$ is drawn from *N*, then (a) according to the central limit theorem [13] and the law of large numbers [14], $X^{(i)}$ will follow the normal distribution with mean, μ and variance, σ^2 that is, $X^{(i)} \sim N(\mu, \sigma^2)$ and (b) the $X^{(i)}$ with the minimum accuracy measure to be considered as a *Bellwether* can then be used to successively predict the remaining samples, $X^{(j)} \; \forall \; j \neq i$. That is the i^{th} *Bellwether* sample predicts the remaining j^{th} samples.

In order to confirm the existence of a potential *Bellwether* from a given population set *N*, empirical analysis was performed based on statistical stratification of *N* using the *X-means* clustering [15], and samples were drawn from each stratum. We realized that, the best population stratum in which potential samples can be drawn from was of size not less than 100 projects. This was empirically validated by applying the *Bellwether method* (in Section 4) to the population set.

2) If P denotes a regular transition probability matrix (TPM) of a Markov chain, then there exist an ergodic Markov chain (EMC) whose respective window can be used as the Bellwether.

Proof: We define the Markov chain as a set of random variables $\{X^{(1)}, \ldots, X^{(t)}\}$ or a collection of stochastic events $\{X(t) | t \geq 0\}$ whereby given the present event of a state at time *t*, the prediction of a future event at *t+1* is independent of past events but the present event. The *iid* sample space for all states at their respective times can be described as a Markov chain if

$$P(X_{t+1} = i_{t+1} | X_t = i_t, X_{t-1} = i_{t-1} \ldots X_1 = i_1, X_0 = i_0) = P(X_{t+1} = i_{t+1} | X_t = i_t) = p_{ij}^{(k)} \quad (1)$$

Assume there exist a variable '*P*' that can be formulated as a matrix of transition probabilities of a Markov chain [6] from

the sample space $\{X^{(1)},\ldots,X^{(t)}\}$ where $i \epsilon T$ denotes the transition states (project ages), then the ij^{th} element, $p^{(k)}_{ij} \epsilon P$ is the probability that the Markov chain starting from a particular state, t_i will transition to t_j after k steps. If $p^{(k)}_{ij}$ is homogeneous, then (2) holds and there exist a unique probability matrix, $\boldsymbol{\theta}_u$ that can form the ergodic Markov chain (EMC) such that for any $\boldsymbol{\theta}_o$ and for large values of u, (3) can be defined as follows:

$$P(\mathrm{X}^{(\mathrm{k})} = j \,|\, \mathrm{X}^{(\mathrm{k}-1)} = i) = \mathrm{p}_{ij}^{(k)} \quad (2)$$

$$\lim_{u\to\infty} \theta_{u+1} = P^u \theta_1 \quad (3)$$

Thus, the EMC can be obtained given a non-negative power of P by making all the entries of the probability matrix non-zero and irreducible. If the limiting state probability matrix (EMC) exists from the TPM constructed with the moving window sample, then that particular sample can be used as the *Bellwether moving window*. We consider such moving window sample as *Bellwether* since its limiting or stationary distribution has been reached and hence can form a potential training set for modeling given it has the best prediction accuracy performance.

3) Given a Bellwether whose ergodic Markov chain is known, then its size and age can be defined.

Proof: Let a sample X with projects $\{p_1,\ldots,p_n\}$ be classified into t states based on their respective ages. Assume that $p_i \epsilon X$ are sorted in non-decreasing order based on the project ages and the EMC of X is known. Then, the age of the *Bellwether* sample, X can be found as the difference between the maximum and minimum states (ages) of the sorted p_i whose EMC is known. Similarly, the size of the *Bellwether* can be found as the total number of projects in X.

A. Assumptions

The following assumptions are made for the effective functioning of the proposed *Bellwether method*:

- Each project used in modeling has an age. Thus, we pruned off irrelevant projects without start and completion dates prior to modeling.
- All selected projects from N have the same features.
- All selected projects are from a single industry.
- All selected projects share common development policies or characteristics.
- Each p_{ij} element in the TPM lies within 0 and 1.
- The sum of each i^{th} row of TPM should be approximately 1.
- All entries of the ergodic Markov chain are non-zero.

4. Bellwether Moving Window: A New Approach

In order to support the replication of this study, we describe the *Bellwether method* for obtaining the moving window in this section. Thus, we introduce a stratification and sampling approach that will assist a software practitioner to select the best moving window from a repository, D of chronological projects to aid in estimating the software effort of a *new* project. We illustrate the general architecture of the *Bellwether method* in Figure 1. Using historical data from D, the following three operators (*SORT+CLUSTER*, *GENERATE TPM* and *APPLY*) can be applied by the software practitioner:

SORT + CLUSTER

Given a set of N completed projects from D, sort based on the project completion dates and stratify the data into q clusters.

1. For all N projects from D, sort in an increasing order using their respective project completion dates.
2. Subject sorted data to *X-means*[1] clustering algorithm [15] to obtain q clusters. Perform data stratification based on the q obtained. That is, stratify N into q clusters whereby each cluster (or window) has a set of chronologically arranged projects.
3. For each window, compute the weighted moving window by applying the respective weighting functions (Triangular, Epanechnikov, Gaussian and Rectangular [10]) on the q windows. We define the resulting q weighted windows in an increasing order as $w_1, w_2 \ldots w_q$.
4. Use w_q as a baseline. w_q contains recently completed projects in a chronological order and can form a *baseline weighted moving window*.

GENERATE TPM

Generate the transition probability matrix (TPM) for the resulting weighted moving window and find the respective ergodic Markov chain (EMC).

5. For the resulting baseline window (w_q), generate the Transition Probability Matrix (TPM) of the Markov chain. Here, use the project ages in w_q as the transition states with their respective effort to generate the TPM.
6. Compute the EMC for the generated TPM in order to validate the limiting (or stationary) distribution of w_q. Thus, using the TPM with s states, successfully perform the squaring of TPM until TPM reaches its stationary distribution (ergodic). That is, when individual probability elements, p_{ij} of TPM cannot be reduced further, we say that TPM is regular or ergodic [6].
7. Report w_q* as a *Bellwether moving window* if the weighted moving window, w_q is stationary and provides the best prediction accuracy for majority of the remaining windows, $w_1, w_2 \ldots w_{q-1}$. Else go to step 4 to update w_q with additional project(s) from w_{q-1} and repeat steps 5 and 6. Conversely, update w_q by sequentially removing project(s) from w_q and adding to w_{q-1} and repeating steps 5 and 6.

APPLY

Apply the Bellwether moving window, w_q to the new project data (set as a hold-out) whose software effort is to be estimated.*

8. Once w_q* is obtained as a *Bellwether moving window* in step 7 prior to its application to the *new* project data (set as a hold-out), its size and age can be defined from its dimensions. The size of the moving window is computed as the total number of projects within the

[1] *X-means* automatically estimate the number of optimal clusters with respect to the Bayesian Information Criterion [15]

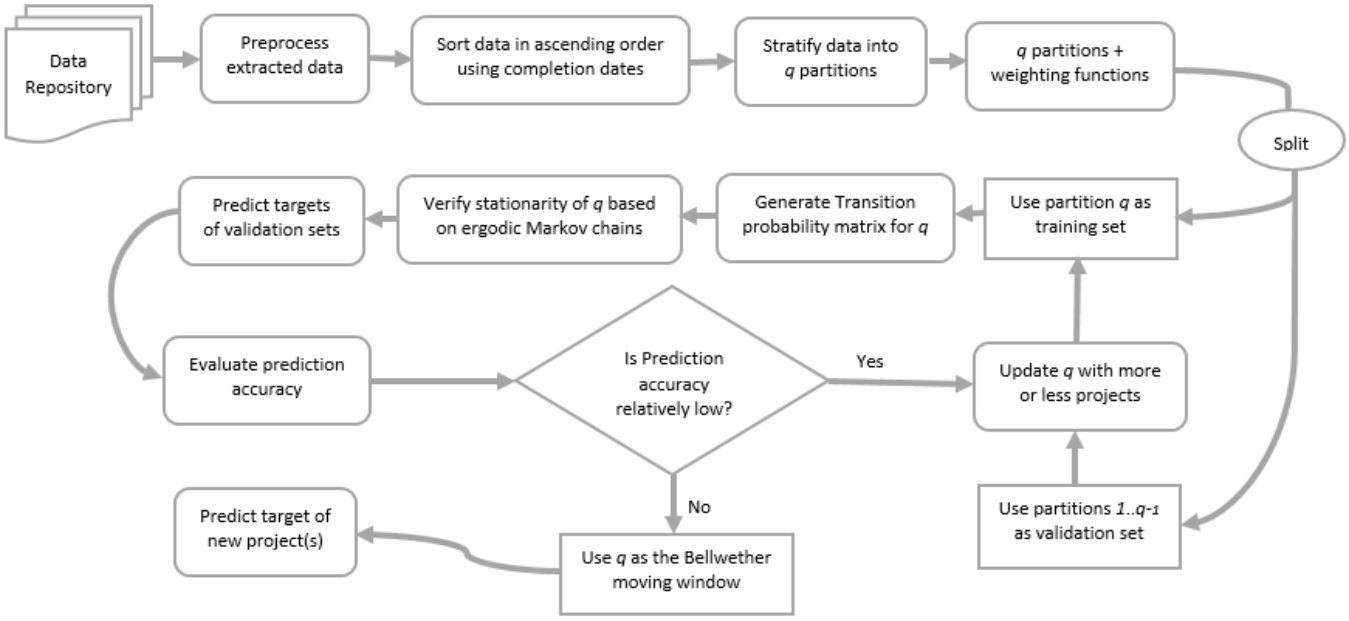


Figure 1. Overview of Bellwether Method for sampling the Bellwether Moving Window

Bellwether moving window whiles the age of the moving window is computed as the difference between the maximum and minimum project ages.

9. Predict the software effort of the *new* project(s) using the resulting *Bellwether moving window* model.

5. METHODOLOGY

A. Dataset

In this study, we use the International Software Benchmarking Standards Group (ISBSG) dataset release 10 which is sourced from the ISBSG dataset repository[2]. It is a chronological cross company dataset with a total of 4106 projects. All historical projects span approximately 20 years between May, 1988 and November, 2007. For easy comparison of our approach with previous moving window studies, we use the same ISBSG dataset that has previously been considered for moving windows [2][5][7]. Although the dataset seems old, the focus is not on how old the dataset is but to prove that *new* projects can be predicted from previously completed projects based on the *Bellwether effect* and Markov chain modeling.

We preprocess the ISBSG dataset following a similar approach by previous studies [2][7]. The preprocessing resulted in a total of 1097 projects (26.7%) selected from the population set (4106 projects). The ISBSG dataset contains variants of projects from different industries with different policies and development methodologies. Hence, such cross projects are heterogeneous in nature and do not share common development policies. We therefore extracted projects from a single industry and hence selected 237 projects (21.6%) out of the total 1097 preprocessed projects. Ten projects (4.2%) with recent completion dates are set aside as hold-out for testing the prediction accuracy of the *Bellwether method*. The selected features from the ISBSG dataset are the size of the projects measured in Unadjusted Function Points (UFP), primary language type (3GL, 4GL), development type (new development, re-development and enhancement), platform (PC, mainframe, midrange and multi-platform), industry sector (manufacturing, banking, insurance and other) and the effort measured in person-hours. It should be noted that, we used the same features considered in previous moving window studies [2][7] to support the replication of previous studies.

B. Experimental Setup and Evaluation Measures

This study employs a proposed *Bellwether method* described in Section 4 to select the *Bellwether moving window* with the respective window size and window age for estimating the software effort of *new* projects. Comparison is made across the use of weighted moving windows (Triangular, Epanechnikov and Gaussian), unweighted moving windows (Rectangular) and growing portfolio (use of all preprocessed projects as training set). These weighting functions have been considered in previous studies [2][10][18]. We evaluate the prediction accuracy performance of the *Bellwether moving window* by using the selected window sample to perform successive predictions on the remaining unselected windows (see detailed explanation in Section 4). Thus, whiles the selected moving window is used as the training set, the unselected windows are used as the validation sets. Thus, after obtaining the *Bellwether* (partition sample with the best prediction accuracy on the remaining partitions), it is then used for estimating the *new* projects. On the other hand, we use the *growing portfolio* as the training set as done in previous studies [2][7][10] for estimation purposes. We went a step further to apply the weighting functions considered in [10] on the *growing portfolio* to empirically investigate their performance in the estimation process. Lastly, we compare our *Bellwether moving window* approach with the *growing portfolio* approach.

[2] http://www.isbsg.org

The estimation model considered in this study is the Automatically Transformed Linear Model (ATLM) proposed by Whigham et al. [3]. ATLM was developed using multiple linear regression with a minimal threshold to act as a baseline for software effort estimation. The prediction accuracy of ATLM proved superior to ensemble methods and a hybrid model (particle swarm optimization, analogy-based estimation and clustering).

We employed the Mean Absolute Error (MAE) which has been proven reliable by Foss et al. [9] and considered in previous studies [3][10][2]. MAE is a risk function that measures the average absolute deviation of the estimated effort values from the true effort values. It provides an unbiased measure that favors model generalization. A robust statistical test called the Welch *t-test* statistic as recommended by Kitchenham et al. [19] is also used to determine the statistical pairwise differences among the weighted (Triangular, Epanechnikov and Gaussian) and unweighted (Rectangular) windows [10] and the growing portfolio. We considered the Kruskal-Wallis *H-test* statistic [4] to find the existence of statistical difference among the four weighting functions applied for both the moving window and the growing portfolio. The Glass's Δ effect size [16] is used to find the practical significance among the *Bellwether moving window* and the *growing portfolio*. We considered these performance measures due to their robustness to outliers.

6. RESULTS

In this section, we present the empirical analysis of the *Bellwether effect* in the chronological dataset. We compare the evaluation performance of using the *Bellwether moving window* and the *growing portfolio* for prediction purposes.

A. Bellwether Effect in Chronological dataset

To show that there exist exemplary projects to be considered as the *Bellwether* sample (postulation 1 in Section 3), we randomly sampled projects from size 30 to the total number of utilized projects (237). Each size ($N \geq 30$) acted as the sample space and the *Bellwether method* (Section 4) was applied on each sample space until a desired threshold size was obtained for the *Bellwether*. Results show that *Bellwether effect* exist in a sample space of $N > 100$ projects to obtain a potential *Bellwether moving window*.

After obtaining the threshold for the existence of *Bellwether*, we report the empirical analysis for this study using the maximum threshold of $N = 237$ (setting 10 projects as hold-out). Results show that, after subjecting the sorted chronological dataset to the *X-means* clustering algorithm, 3 clusters (with initial approximate size of 76 projects) were recorded. The transition probability matrix (TPM) from the Markov chain modeling show that, there exist an ergodic Markov chain (EMC) from the 3 clusters or windows whereby its respective window can be considered as the *baseline window*. We empirically validated this initial *baseline* with MAE and updated the sizes (as shown in step 7 of Section 4) until the *Bellwether moving window* was obtained.

We present the prediction evaluation performances of different potential *Bellwether moving windows* in Figure 2. Results from Figure 2 depict the MAE evaluation of using relevant windows weighted with the four functions (Gaussian, Triangular, Epanechnikov and Rectangular). The selected

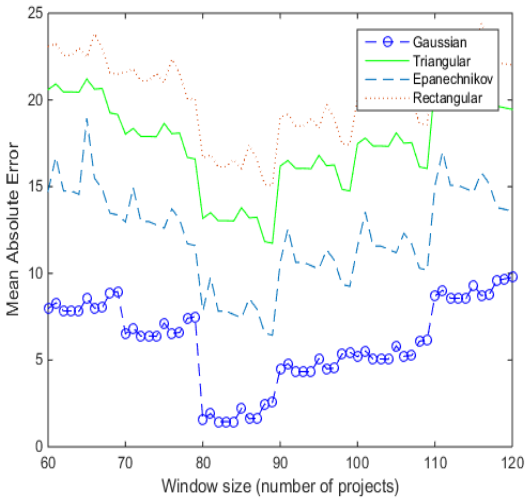


Figure 2. MAE prediction evaluation of different window sizes subjected to the four weighting functions

potential *Bellwether moving windows* were used to predict the projects that were set as hold-out as shown in Section 4. We present results of potential *Bellwether moving windows* of sizes ranging from 60 to 120 since they yielded relatively minimal error evaluation measures on average. On average the best number of projects to be considered as the *Bellwether moving window* ranges from 80 to 90 projects (35.2% to 39.7% of the training data) irrespective of the weighting function considered. These projects have project ages of not more than 3 years. Results also show that the Gaussian weighting function yielded the best relative predication accuracy as compared to the remaining weighting functions (Figure 2).

B. Best Sizing and Aging for Bellwether moving window

Results from Table 1 show the window sizes and ages of the potential *Bellwether moving windows* with their respective performance measures. As a result of establishing the baseline that the Gaussian weighting function yields a relative prediction performance, we investigated which window size and age yielded the best prediction accuracy. Out of the range of 80 to 90 exemplary projects, we realized from their respective MAE that a subset of 82 to 84 projects (36.2% to 37.0% of the training data i.e. 227 projects) with their respective ages of 1.5 to 2 years yielded a relative prediction accuracy (Table 1).

Table 1: Prediction Accuracy Performance of Bellwether moving windows (exemplary projects) with respect to MAE

Window age (years)	Window size	MAE
1	80	1.5868
1	81	1.9144
1.5	82	**1.4523***
1.5	83	**1.4546***
2	84	**1.4377***
2	85	2.2069
2	86	1.6152
2	87	1.6517
2.5	88	2.4752
3	89	2.5686

'' used to indicate best prediction performance results (comparison made for each evaluation measure along each column). Thus, the minimum MAE signifies the best prediction accuracy.*

C. Statistical Comparison of the Bellwether moving window versus the Growing portfolio

To investigate the significance of the estimation accuracy performance of the *Bellwether moving window* models, we performed a pairwise statistical test to benchmark the *Bellwether moving window* models with the *growing portfolio* models. The Welch *t-test* result show that at 5% asymptotic significance level, there is a significant difference between the *Bellwether moving window* and the *growing portfolio*. This statistical significance is confirmed by the large Glass's Δ effect size value of 0.627 (>0.5) implying practical significance in the use of the *Bellwether moving window* over the *growing portfolio*.

D. Effect of weighting functions on the Bellwether Moving windows

The Kruskal-Wallis *H-test* resulted in a Chi-square value of 71.17 (*p-value<0.05*) showing the existence of statistical significant differences at 5% asymptotic significance level across the weighting functions for the *Bellwether moving windows* (Triangular, Epanechnikov and Gaussian). Similarly, incorporating the unweighted moving window (Rectangular) [10] with the weighted *Bellwether moving windows* resulted in a Chi-square value of 97.52 (*p-value<0.05*) showing the existence of statistical significant differences. This confirms result by a previous study [2] that the use of a weighting function can affect the estimation performance of a prediction model. We realized that the use of the Gaussian weighting function on the *Bellwether moving window* improves the relative prediction accuracy performance.

7. Threat to Validity

We used a single release of the ISBSG dataset (release 10) as considered in previous studies [2][7]. This dataset is a convenience sample but cannot be a general representation of all chronological datasets. Therefore, our results might not be confidently generalized beyond this dataset.

8. Conclusion and Future work

The findings show that, *Bellwether effect* exist in software effort estimation and its respective *Bellwether moving window* has a relative effective window size of 82 to 84 projects (36.2% to 37.0%) and window age of 1.5 to 2 years for improved prediction accuracy. Incorporating weighting functions on the *Bellwether moving window* affect the prediction accuracy when the window size and window age of the *Bellwether moving window* differs from the aforementioned threshold. We observed that, the Gaussian weighting function was more advantageous for software effort estimation when using moving windows. The *Bellwether moving window* will continue to be useful for prediction purposes provided the phenomenon being measured share similar characteristics with the *Bellwether* projects.

The introduced *Bellwether method* can be used in other software engineering domains for selecting the *Bellwether moving window* to be considered for the training and validation needs of predictive models. We intend to extend the *Bellwether method* to empirically analyze the Finnish dataset and other industrial projects to further validate the existence of *Bellwether moving windows.*

Acknowledgment

This work is supported in part by the General Research Fund of the Research Grants Council of Hong Kong (No. 125113, 11200015 and 11214116), and the research funds of City University of Hong Kong (No. 7004683 and 7004474).

References

[1] R. Krishna, T. Menzies, and W. Fu. "Too much automation? the Bellwether effect and its implications for transfer learning." *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2016.

[2] S., Amasaki, and C. Lokan. "On the effectiveness of weighted moving windows: Experiment on linear regression based software effort estimation."*Journal of Software: Evolution and Process* 27.7 (2015): 488-507.

[3] P. A. Whigham, C. A. Owen, and S. G. Macdonell. "A baseline model for software effort estimation." *ACM Transactions on Software Engineering and Methodology (TOSEM)* 24.3 (2015): 20.

[4] T. W. MacFarland, and J. M. Yates. "Kruskal–Wallis H-Test for Oneway Analysis of Variance (ANOVA) by Ranks." *Introduction to Nonparametric Statistics for the Biological Sciences Using R*. Springer International Publishing, 2016. 177-211.

[5] C. Lokan, and E. Mendes. "Applying moving windows to software effort estimation." *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE Computer Society, 2009.

[6] R. P. Dobrow. "Markov Chain Monte Carlo." *Introduction to Stochastic Processes With R* . Wiley Online Library (2016): 181-222.

[7] C. Lokan, and E. Mendes. "Investigating the use of duration-based moving windows to improve software effort prediction: A replicated study."*Information and Software Technology* 56.9 (2014): 1063-1075.

[8] B. Zhou, H. Okamura, and T. Dohi. "Enhancing performance of random testing through Markov chain Monte Carlo methods." *IEEE Transactions on Computers* 62.1 (2013): 186-192.

[9] T. Foss, E. Stensrud and B. Kitchenham. "A simulation study of the model evaluation criterion MMRE." *IEEE Transactions on Software Engineering* 29.11 (2003): 985-995.

[10] S. Amasaki, and C. Lokan. "A replication study on the effects of weighted moving windows for software effort estimation." *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2016.

[11] Z. Tu, and S. C. Zhu. "Image segmentation by data-driven Markov chain Monte Carlo." *IEEE Transactions on pattern analysis and machine intelligence* 24.5 (2002): 657-673.

[12] C. Lokan, and E. Mendes. "Investigating the use of duration-based moving windows to improve software effort prediction." *2012 19th Asia-Pacific Software Engineering Conference*. Vol. 1. IEEE, 2012.

[13] H. Fischer. *A history of the central limit theorem: From classical to modern probability theory*. Springer Science & Business Media, 2010.

[14] K. Yao, and J. Gao. "Law of large numbers for uncertain random variables." *IEEE Transactions on Fuzzy Systems* 24.3 (2016): 615-621.

[15] D. Pelleg, and A. W. Moore. "X-means: Extending K-means with Efficient Estimation of the Number of Clusters." *ICML*. Vol. 1. 2000.

[16] M. Shepperd, and S. MacDonell. "Evaluating prediction systems in software project estimation." *Information and Software Technology* 54.8 (2012): 820-827.

[17] S. Amasaki, and C. Lokan. "The effects of gradual weighting on duration-based moving windows for software effort estimation." *International Conference on Product-Focused Software Process Improvement*. Springer International Publishing, 2014.

[18] S. Amasaki, and C. Lokan. "The evaluation of weighted moving windows for software effort estimation." *International Conf. on Product Focused Software Process Improvement*.Springer Berlin Heidelberg, 2013.

[19] B. Kitchenham, L. Madeyski, D. Budgen, J. Keung, P. Brereton, et al. "Robust statistical methods for empirical software engineering." *Empirical Software Engineering* (2016): 1-52.

Security Requirements for Tolerating Security Failures

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

Abstract— This paper describes security failure-tolerant requirements, which tolerate the failures of security services that protect applications from security attacks. A security service, such as authentication, confidentiality or integrity security service, can be always broken down as advanced attack skills are coined. There is no security service that is forever secure. This paper describes an approach to developing the security failure-tolerant use case that specifies the security requirements for tolerating the breaches of security services. A security failure-tolerant use case is modeled along with application use case and security use case, and specified with application use case description. Threats to applications are identified and modeled to develop security failure-tolerant requirements. Online shopping system is used for illustrating security failure-tolerant requirements.

Keywords - Security Requirements; Security Failure-Tolerant use case; Security use case; Application use case

I. INTRODUCTION

Secure applications are designed with the security services that are made to achieve security goals, such as authentication, authorization, confidentiality, integrity, availability and non-repudiation. The security services seem to be unbreakable enough to protect security assets in the applications from attacks. However, in reality, although applications are designed with unbreakable security services, the security services are always broken down as attack skills are getting crafty [2, 15]. To make applications more secure, it is necessary for security services to be tolerated when they are broken down.

Several approaches [7, 8, 9, 10, 11] were developed to make applications secure in software development. Most of the approaches have focused on specifying and designing applications with security services in order to make applications secure. Security requirements are specified with Unified Modeling Language (UML) [1] and its extended notation [7, 8], separately from application requirements [10]. Secure software architecture is designed using secure connectors [11] that encapsulate security services. However, less attention has been paid to the tolerance of broken security services in terms of security requirements for secure applications.

This paper describes an approach to developing security failure-tolerant requirements that tolerate broken security services. The tolerance of breached security services is specified on the assumption that any security services can be broken down. The security failure-tolerant approach aims at reducing the possibility of security damage to security assets in the applications from the breaches of security services. The security failure-tolerant approach is adopted from fault-tolerant approach to minimizing the system damage from a fault of systems. The proposed approach can delay attacks until the security failure-tolerant use cases are compromised. Although the security failure-tolerant approach might not be the ultimate solution to security, it can be a solution to make applications more secure by mitigating the breaches of security services.

II. RELATED WORK

Threat Modeling. Threats in a system have been modeled by several approaches, which include attack trees [2], data flow diagrams [3], and UML-based modeling [4, 5, 6]. Attack trees in [2] provide an approach to modeling and analyzing the threats of systems, and the threats are analyzed in terms of attacker's capabilities. The design models in [3] are specified with data flow diagram, 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 [4], abuse cases [5], and HAZOP (Hazard and Operability Analysis) [6], have been developed for object-oriented software systems. The approaches model threats using the use case model in UML and capture security requirements for them.

Secure Software Development. Several researches for developing secure software have been done in terms of secure requirements and design. The studies in [7, 8] proposed a new modeling language based on UML for the model-driven development of secure, distributed systems. The research in [17] illustrates an ontology-based approach that uses predefined pattern-based templates to aid requirements engineers in the formulation of security requirements. Security patterns in [9] address the broad range of security issues that should be taken into account in the stages of software development lifecycle.

Mitigation of Security Failures. Security failures can be mitigated by several approaches, such as layered security (defense in depth) [12], intrusion tolerance [16], and self-protection [13]. The layered security [12] 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.

DOI reference number: 10.18293/SEKE2017-098

III. SECURITY REQUIREMENTS FOR SECURITY FAILURE-TOLERANCE

A. Threat Modeling

Threats to a security failure-tolerant application focus on the security assets in the application that should be protected from attacks. A security asset can be a security relevant input to applications, secure data maintained in an application, and the system itself on which an application is running [4]. The security relevant input to applications is a user's input to applications or the input from an external system or external devices to applications in which the inputs require security. An account identification (ID) or password entered by a user to an application can be an example of the security relevant input to applications. A secure data stored in an application can be a target of an attack. The example of a secure data can be the credit card information maintained by an electronic commerce application or a patient's medical record 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 an application's availability.

The security assets in a security failure-tolerant application can be identified by analyzing the use case descriptions for each application use case. The use case description describes application business logic in terms of the actor's inputs to a system and the system's responses to the actor. Also, a use case description addresses the data stored in the application and the actions applied to the data so as to process the actor's input and generate a response to the actor. The actor's input or the data stored in an application is a security asset if it requires security.

The *make order request* use case in online shopping application [14] receives a customer order request, checking the sufficient credit to pay for the requested items and creates a delivery order for the customer if the credit is sufficient. The use case description for *make order request* use case is described as follows:

Use case name: Make Order Request
Summary: Customer enters an order request to purchase items. The customer's credit card is checked for validity and sufficient credit to pay for the requested items.
Actor: Customer
Precondition: Customer has selected one or more catalog items.
Main sequence:

1. Customer selects the order request service.
   <Secure ID and Password>
2. System prompts the input for order request to customer.
3. Customer provides a purchase order request and customer account ID and password to pay for the purchase **<Threat point: ID and Password>**.
   <Tolerant ID and Password>
   <Secure Credit Card>
   <Tolerant Credit Card>
4. System retrieves customer account information, including the customer's credit card details **<Threat point: Credit Card>**.
5. System checks the customer's credit card for the purchase amount and, if approved, creates a credit card purchase authorization number.
6. System creates a delivery order containing order details, customer ID, and credit card authorization number.
7. System confirms approval of purchase and displays order information to customer.
8. System sends email confirmation to customer.

Alternative sequences:
Step 4: If customer does not have an account, the system prompts the customer to provide information in order to create a new account.
Step 5: If authorization of the customer's credit card is denied, the system prompts the customer to enter a different credit card number.
Threat and Security:

- Threat at Step 3: Release ID and Password
  - Security Asset: ID and Password
  - Description: ID and Password can be released to attackers
  - Security goal: Confidentiality of ID and Password
  - Security use case: Check Keystroke Logging security use case
  - Security failure-tolerant use case: Verify Image security failure-tolerant use case
- Threat at Step 4: Release Credit Card
  - Security Asset: Credit Card
  - Description: Customer credit card information might be released
  - Security goal: Confidentiality of Credit Card
  - Security use case: Check Malicious Code security use case
  - Security failure-tolerant use case: Fraud Monitor security failure-tolerance use case

Post-condition: System has created a delivery order for the customer.

The customer account ID and password at step 3 in the *make order request* use case description can be a security asset in terms of a customer input that requires security. Also, the customer's credit card details at step 4 are another security asset stored in the application so that the system processes the customer's purchase request.

A threat identified is modeled with application use cases in the use case model. A threat threatens an application use case at a threat point. In this paper, a threat is represented using the use case notation in the use case model, but a threat use case does not have a specific actor because an attacker can be any malicious person. Also the threat use case does not have a common scenario as to how to realize the threat. This is because an attacker can realize a threat in an unpredictable way. A threat point is a point in the application use case where a threat can occur. Also, a threat point is a step in the use case description for an application use case where a security asset is jeopardized if a security service is broken and there is no any security failure-tolerant service to protect the asset.

The threats to *make order request* use case are modeled in Fig. 1 in which the *release ID and password* threat threatens the *make order request* use case at the *ID and password* threat point. Similarly, the *release credit card* threat threatens the *make order request* use case at the *credit card* threat point. A threat point is designated in the use case description by means of <threat point> with the threat point name. The *ID and Password* threat point is designated at step 3 as <threat point: ID and Password> in the *make order request* use case description. Also the *credit card* threat point is presented with <threat point: Credit Card> at step 4 in the same use case description.

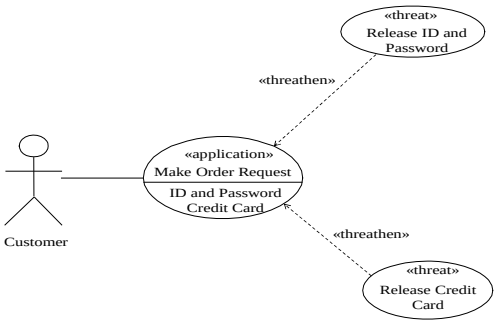


Fig. 1 Threats to Make Order Request application use case

Each threat is specified to analyze security concerns so that a security failure-tolerant service is developed along with a security service. A threat can be described in the use case description for an application use case. A threat is described shortly in the threat and security section of the use case description in terms of threat name, security asset, threat description, and security goal. In the *make order request* use case description above, the release ID and password threat is specified with security asset (ID and password), description (ID and Password can be released to attackers), and security goal (Confidentiality of ID and Password). Similarly, the release credit card threat is specified in the use case description. As 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 [15].

B. Security Requirements Modeling

Security requirements of security services for an application system are specified with security use cases [10] separately from non-secure application use cases. When the 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 application requires the security use case. An application use case provides an extension point where a security use case extends the application use case.

The security use cases for the *make order request* application use case are depicted in Fig. 2 in which the *check keystroke logging* security use case and *check malicious code* security use case are provided for the non-secure *make order request* application use case. A user's computer might be infected with malicious keystroke logging code that records user credentials and sends them to a third party location to do further harm. The *check keystroke logging* security use case mitigates the leak of user's ID and password using anti-malware software on the user's computer. The *make order request* application use case is extended to the *check keystroke logging* security use case at the *secure ID and password* extension point if the application use case requires the security use case. The *secure ID and password* extension point is described in the use case description for *make order request* application use case. The check keystroke logging security use case is specified as follows:

Security use case: Check Keystroke Logging
Summary: System checks a keystroke logging attack to protect customer input.
Actor:
Precondition: Anti-Keystroke Logging software is running.
Description:
1. System checks a keystroke logging attack.
2. If system detects a keystroke logging software, system displays a warning message "Keystroke Logging Attack" and removes the keystroke logging software.
3. System logs a keystroke logging attack.

Alternatives:
Post-condition: keystroke logging software has been checked.

The *check malicious code* security use case is another security measure that has been employed to protect the *make order request* application use case. Malicious code may get in the application system and it can release user's credit card information to an attacker. When the *make order request* use case requires the *check malicious code* use case, the *check malicious code* security use case is extended from the *make order request* application use case at the *secure credit card* extension point, which is designated in the *make order request* application use case.

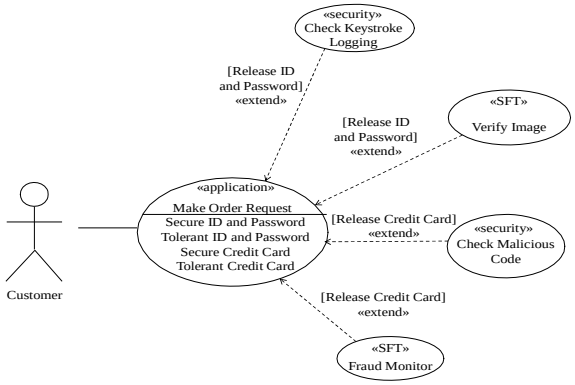


Fig. 2 Security use cases for make order request use case

C. Security Failure-Tolerant Requirements Modeling

Security failure-tolerant requirements are modeled with security failure-tolerant use cases, which tolerate the breaches of security services for applications. By careful separation of concerns, the security failure-tolerant requirements are captured in security failure-tolerant use cases separately from security use cases and application use cases. When an application use case requires a security failure-tolerant use case, the security failure-tolerant use case tolerates the breach of security use case.

Fig. 2 depicts *verify image* and *fraud monitor* security failure-tolerant use cases, which tolerate the breaches of *check keystroke logging* and *check malicious code* security use cases for *make order request* application use case, respectively. The *verify image* security failure-tolerant use case verifies that an image selected by the 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 should know of the customer's image registered in the system in order to make a malicious purchase order. The *fraud monitor* security

failure-tolerant use case activates a service to monitor credit card fraud so that it prevents the damage caused by the release of credit card information. Malicious code hidden in the system might release the customer's credit card information to an attacker if the *check malicious code* security service fails to detect malicious code. However, the *fraud monitor* security failure-tolerant use case tolerates the attacker's fraud of released credit card. Providing a credit monitoring service to its customers can ensure that even if credit card information gets released, customer will be protected from further damage.

A security failure-tolerant use case is extended from an application use case at an extension point if the application requires tolerating the breaches of security service. An extension point for a security failure-tolerant use case is a location in an application use case where the security failure-tolerant use case extends the application use case. An extension point of a security failure-tolerant use case is distinguished from that of a security use case. For example, the *verify image* security failure-tolerant use case extends the *make order request* use case at the *tolerant ID and password* extension point (Fig 2), where the *check keystroke logging* security use case is extended from the *make order request* use case at the *secure ID and password* extension point (Fig. 2). The *tolerant ID and password* extension point is designated in the *make order request* use case description. The *verify image* security failure-tolerant use case is described as follows:

Tolerant use case: Verify Image
Summary: Customer clicks an image rather than keystroking his/her ID and password and system verifies the image.
Actor: Customer
Precondition: Customer's personal image is stored in the system.
Description:
1. System displays multiple images, which includes the image that customer has selected while registering for the system.
2. Customer selects an image that he/she has selected when registering for the system.
3. System verifies that the image selected by the customer is matched with the customer's image stored in the system.
4. If the images are the same, system approves that the customer makes an order.

Alternatives:
Step 4: If the customer selects the incorrect image consecutively for 2 times, the customer account is locked.
Post-condition: System has verified an image selected by a customer.

IV. Conclusions and Future Work

This paper presumes that security services are broken all the time in a real-world setting. On this assumption, first our approach has identified threats associated with security assets in terms of security relevant user's input, secure data stored in the application, and the system on which an application is running. Second we constructed security use cases against the threats so that the application would be protected from the threats identified. Finally, security failure-tolerant use cases have been specified to tolerate the breaches of security use cases.

The security failure-tolerance can be envisioned with further research. The security failure-tolerant requirements can be extended to security failure-tolerant analysis modeling that describes the static modeling and dynamic modeling. Also, this research can be extended to develop a framework for security failure-tolerant requirements in which security failure-tolerant use cases are categorized with security use cases in terms of security goals.

References

[1] J. Rumbaugh, G. Booch, and I. Jacobson, "The Unified Modeling Language Reference Manual (2nd Edition)," Addison Wesley, Reading MA, 2004.

[2] B. Schneier, "Attack trees: Modeling security threats," Dr.Dobbs Journal, pages 21–29, December 1999.

[3] M. Abi-Antoun, D. Wang and P. Torr, "Checking Threat Modeling Data Flow Diagrams for Implementation Conformance and Security", ASE 2007, 21 pages, 2006.

[4] G. Sindre and L. Opdahl, "Eliciting Security Requirements with Misuse Cases," Requirements Engineering, Volume 10 Issue 1, January 2005, pp. 34 - 44.

[5] 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.

[6] T. Srivatanakul, "Security Analysis with Deviational Techniques," PhD thesis, Department of Computer Science, University of York, UK, 2005.

[7] 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.

[8] J. Jürjens, "UMLsec: Extending UML for Secure Systems Development", Fifth International Conference on the Unified Modeling Language, London, UK, 2002.

[9] E. B. Fernandez, "Security Patterns in Practice", Wiley, 2013.

[10] 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.

[11] 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.

[12] S. Gantz, "Layered Security Architecture: Establishing Authentication, Authorization, and Accountability", securityarchitecture.com/docs/, 2008.

[13] P. Horn, "Autonomic Computing: IBM's Perspective on the State of Information Technology", http://people.scs.carleton.ca/~soma/biosec/readings/autonomic_computing.pdf, 2001.

[14] H. Gomaa, "Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures", Cambridge University Press, 2011.

[15] 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.

[16] I. E. 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.

[17] 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.

SemHunt: Identifying Vulnerability Type with Double Validation in Binary Code

Yao Li, Weiyang Xu, Yong Tang, Xianya Mi, and Baosheng Wang
College of Computer Science
National University of Defense Technology
Changsha, Hunan, China
{liyao15,xuweiyang11,ytang,mixianya09}@nudt.edu.cn,wangbaosheng@126.com

***Abstract*—when manufacturers release patches, they are usually released as binary executable programs. Vendors generally do not disclose the exact location of the vulnerabilities, even they may conceal some of the vulnerabilities, which is not conducive to study the in-depth situation of security for the need of consumers. In this paper we introduce a vulnerability discover method using machine learning based on patch information - SemHunt. Firstly, we use it to compare two versions of the same program to get the potential vulnerability-patched function pairs using binary comparison technology. Then, we combine it with vulnerability and patch knowledge database to classify these function pairs and identify the possible vulnerable functions and the vulnerability types. We completed a prototype of SemHunt, which can effectively identify vulnerable function types and the location of corresponding vulnerabilities, which are not revealed in the released patch files. Finally, we test some programs containing real-world CWE vulnerabilities, and one of the experimental results about CWE843 shows that the results returned from only searching source program are about twice as much as the results from SemHunt. We can see that using SemHunt can significantly reduce false positive rate of discovering vulnerabilities compared with analyzing source files alone.**

***Index Terms*—software security; binary comparison; vulnerability; patch file; machine learning**

I. Introduction

At present, the search for vulnerable software functions is mainly based on software source code, which means that professional testing tools on source code can find the vulnerabilities through automatic analysis of the code. However, obtaining the source code is a very demanding condition, because many software only provides executable programs such as the large commercial software MS Office and the free off-source software Adobe Reader. In addition, manufactures usually release patches in binary executable program without detailed vulnerability information and its corresponding location. Therefore, finding vulnerabilities in the source code has great limitations. They are not conducive to study the in-depth situation of security for the need of consumers. Now, there are some problems in vulnerability analysis techniques at the binary level. Firstly, the binary comparison technique between two versions of the same program may miss some match functions[1][2][3][4], and the analysis of the vulnerability still needs a lot of manual work. There are some binary comparison tools like BinDiff[5][6], BinSlayer[7] and so on[8], which can get good matching results. They compare two functions to get their similarity, and determine whether the two functions match by setting a threshold. In contrast, our method aims at finding the potential vulnerability-patched function pairs, which requires us to modify the current binary comparison algorithm to adapt to our method.

Another problem is that the search for vulnerability at the source program level may have high false positive rate[9]. They establish the vulnerability knowledge database and find the similar functions to the test ones to determine the vulnerability types. Eschweiler and etc. match the function basic blocks' flow graph to find the pairing vulnerable functions to the test software function[10]; Ming and etc. firstly find the match basic blocks through semantic features, then extend the matched blocks to the functions[1]. These methods only identify the possible vulnerable functions and their vulnerability types at the source program level. Lacking verification from other ways may lead to the high false positive rate.

Our paper firstly presents a novel approach to finding the semantic differences between two versions of the same program, aiming at finding the exact match between the potential vulnerable functions and potential patched functions. Then, we use the machine learning classification algorithms which are combined with the patch information to make a double verification for identifying vulnerability types. Thus, we can increase the accuracy of the vulnerability report, as well as reduce the false positive rate and identify these vulnerable functions types and corresponding locations more effectively, which are not disclosed in detailed information when the patch file is released.

The main contributions of the paper are as follows:

(i) We build a set of common vulnerability functions, as well as a corresponding set of patch-functions, which are used to train the machine-learning classifier.
(ii) We propose a new binary comparison method between the unpatched-vision and patched-vision software, to find out the possible vulnerability-functions and its patch functions.
(iii) We have implemented a prototype of SemHunt.

Our paper will be described in the following order. Section 2 will introduce a whole framework. Section 3 will introduce the key points of the SemHunt. Section 4 will introduce

DOI reference number: 10.18293/SEKE2017-117

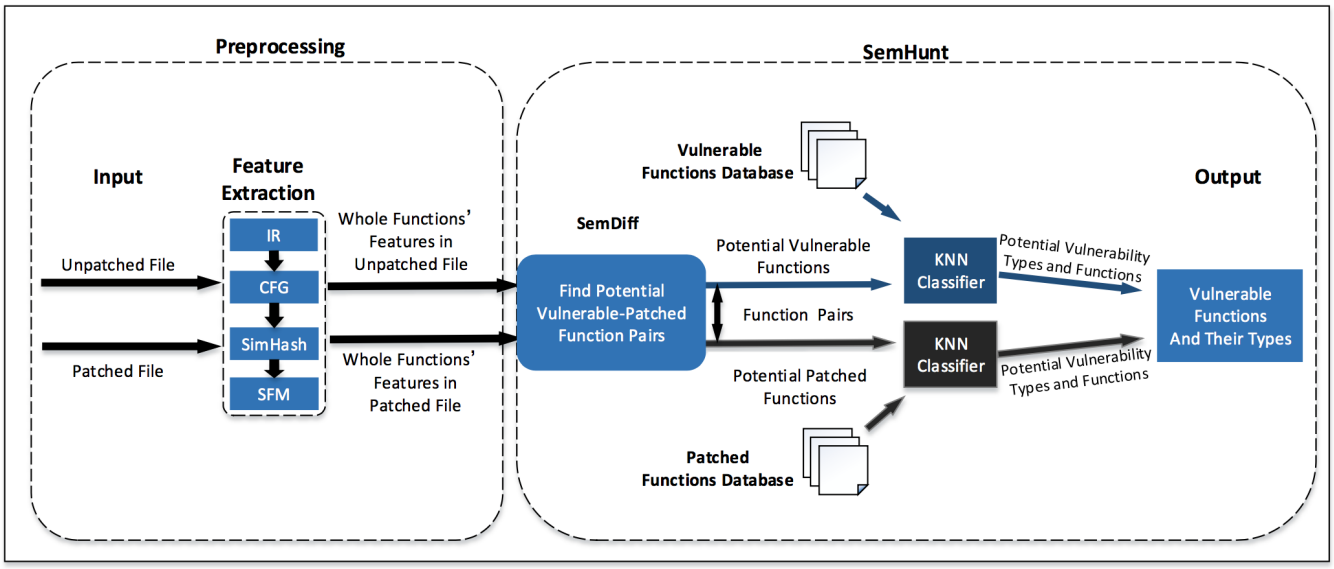


Fig. 1: System Architecture

the experience in detail. Weakness and future work will be introduced in Section 5.

II. System Architecture

Figure 1 shows our overall system structure. Firstly, the two binary files are loaded into our angr platform, and it run its own disassembly tool to generate the assembly code. Then the assembly code are transferred into the intermediate language using the IR converter(VEX), and the CFG is generated through the CFG constructor. We extract the semantics information and digital feature of the function. After that we use them in the binary comparison algorithm to get the possible pairs of unpatched-function and patched-function, we extract the feature and Simhash of each function through the digital feature extractor. We add the match between patched-file and patched-function set to reduce the rate, because only finding the match function between vulnerability-functions set and the unpatched file to get the vulnerability type may have a high false positive rate. Then we put the features into the machine learning classifier to get the corresponding matches. We combine the results from two classifier to get the final potential vulnerability types. Then we will talk about some parts briefly.

III. The Keys Of SemHunt

There are two main parts in the SemHunt: Binary comparison algorithm and Vulnerability Search Combined With Patch Information.

A. Binary comparison algorithm

We modify the BinDiff algorithm to adapt our system. Bindiffs selectors only consider the structural characteristics of the three-tuple information, such as its selector of callgraph, which only includes the numbers of basic blocks, the number of edges and a sub-function call number. The selector for the control flow graph of the basic block contains the number of blocks of the shortest path to the exit of the function, the number of blocks passed from the entry point to the shortest path of the block and the number of points of the sub-function in the basic block. They are all used in a Euclidean space to carry out the minimum distance to judge[5].

However, our SemDiff changes not only in the selector, including the semantic judgments and more digital information, but also in the process of loop. We do not only check the parent-child nodes in one loop (this method may miss a lot of matches), but also carry out their own check. To determine the two functions are functionally identical, we define a rule as follows:

Definition 1: pair functions equivalence formulas of data-Given two list of data that are recorded in the memories and registers: $X = [x_0, x_1, \cdots, x_m]$, and $Y = [y_0, y_1, \cdots, y_n]$. For every x_i in X, there is a bijection, $y_j = f(x_i)$ is existed ,then ,we call the data is same. The formula is described as follows:

$$\forall x_i, \exists y_j \ = \ f(x_i), f(x) \ is \ a \ bijection$$

There are three parts in the SemDiff: WholeMatch, MatchPro and SemDiff. As with the previous algorithms, at the beginning of the initially matching process, we find the functions which are uniquely matched, which means that, we find some functions that have the same symbol expressions and digital features.

At the first stage of the algorithm, the inputs are the function sequences of the two programs respecitively. After the "Whole Match", we get a result of the match function and two lists of functions that are still unmatched.

Then, it is the second stage. We call it "MatchPro", because it is applied in the propagation progress. The input of the

second part is the matched function set, which is got from the first stage and the remaining unmatched function sequences . Then we match each function from a small set, which can be got after the function ”FindPaAndCh”. It is a function that return the set of the parent nodes and children nodes to the functions in the matched set. After we get the subset, we use them into the ”WholeMatch” to get the matched functions until we finish traversing all matched functions.

The ”SemDiff” is a combination of ”WholeMatch” and ”MatchPro”. After we execute the three process, we get a matched-function set and an unmatched-function set. The match set includes the functions that are absolutely the same, so they are not the vulnerable functions and corresponding patched functions. We can get the potential vulnerable-patched function pairs in the unmatched-function set. We compare each function pair in it. If the similarity is over 80%, we consider this pair as the potential vulnerable-patched function pair and put them in the candidate set.

B. Vulnerability Search Combined With Patch Information

In this search stage, we use the machine learning algorithm as our classifier. We will introduce the digital feature extractor first. Each function holds a lot of digital information or metadata, such as the number of instructions and so on. The digital feature extractor is designed to extract a set of digital features, which can represent the binary function. The paper[10] indicates the number of instructions, the size of local variables, the number of parameters, the number of CFG-based blocks and the numbers of edges can be extracted as the digital features. We also classify and record the instructions according to the instruction types, like arithmetic instructions, data transfer instructions, logic operations instructions, function call instructions, function jump instructions and so on. Different from the paper[10], we calculate the SimHash of each function which is added to the digital features. Then we use the machine

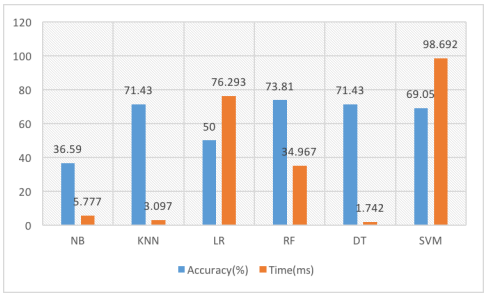


Fig. 2: The Performance of Machine Learning Algorithm

learning algorithms to classify the functions into different types according to the vulnerable-function database and the patched-function database. Considered the performance of each machine learning algorithm(fig2), we finally use the KNN algorithm as our classifier and we modify it to adapt our system. We check the results returned from the classifier. If each result has the similarity over 90% with the test function, then we still put them into the result set, or drop them. In this case, we can adapt to the real situation that not every unmatched-function pair with the similarity over 80% is the vulnerable-patched function pair.

We can also explain the rationality of the low false rate by double validation. Though the digital feature can’t represent a specific function, the results returned from the machine learning classifiers include the true function type. If the true function type is in each result, the intersection must have the true label. Thus, we can reduce the range of the candidate through making an intersection of two results returned from two classifiers respectively.

IV. Experiment

Our experiment is carried out in the system Ubuntu 14.04, running in an angr virtual environment[11][12][13] . The language we use is Python. We firstly introduce the two databases’ construction. Then, we compare some machine learning algorithms performance to get the best one. Then, we modify it to adapt our system. At last, we introduce some true examples which have the CWE vulnerabilities to verify the accuracy and compare performances between double validation and single validation.

A. The Construction of Databases

We have downloaded the common vulnerable functions on CVE website, which contains the corresponding patched function, including Stack Based Buffer Overflow, Heap Based Buffer Overflow, Buffer Underweite, Integer Overflow, Use after free, Double Free, Type Confusion and other vulnerabilities, about totally 148 kinds, more than 4000 vulnerable-functions and more than 5000 patched-functions. After we get the source code, we have to compile them in different platform, here we just consider the Linux and Windows.

Because the instruction confusions have negative influence on the performance, we have to preprocess the data information. According to the feature we extracted and the Simhash, we just focus on instruction rearrangement and register renaming.

B. An Example with CWE843

At first, we take the program with CWE843 and its patched-file as the input respectively in the KNN classifier. Then we can get a list of results which are function-vulnerability pairs. As TABLE I and TABLE II shown:

Function	Type	T/F
plt._libc_start_main	Missing_Handle,· · · ,Infinite_Loop	F
plt.printf	Infinite_Loop,· · · , Reachable_Assertion	F
sub_40124e	Type_Confusion,Null_Deref_From_Return	T
sub_400f80	Type_Confusion,Null_Deref_From_Return	T
sub_4012a0	Type_Confusion	T
sub_400f2e	Type_Confusion	T
· · ·	· · ·	· · ·
_init	Missing_Handle,· · · , Reachable_Assertion	F
_libc_csu_init	Infinite_Loop,· · · , Reachable_Assertion	F

TABLE I: The Final Result of unpatched-program

T represent the true type is in the Type set; F represent the true is not in the Type set.

Function	Type Set	T/F
_init	Mismatched_Memory_Management	F
_libc_csu_init	Infinite_Loop,· · · , Reachable_Assertion	F
sub_40101c	Type_Confusion	T
sub_4010e2	Type_Confusion	T
sub_401175	Type_Confusion,Reachable_Assertion	T
sub_400e57	Type_Confusion	T
· · ·	· · ·	· · ·
printLine	Missing_Handle,· · · , Reachable_Assertion	F
deregister_clones	Uncontrolled_Recursion	F

TABLE II: The Final Result of patched-program

T represent the true type is in the Type set; F represent the true is not in the Type set.

The vulnerable program has 51 functions totally, and the classifier returns 16 functions that may be the potential vulnerability function. However only 4 functions have bugs. Also, there are 51 functions in the patch file with 4 bug functions, however 18 functions are returned from the classifier. Through observing the data, we find that there are nine functions are same which are _init, _libc_csu_init,· · · , printLine. They cost us too much energy to analyse, however they can't be the vulnerable functions.

Then, we put the two test programs into the SemHunt. We get a list of functions which have the similarity over 80% from the unmatched set. Thus we get the candidates that may be the vulnerable-function and patched-function pair as TABLE III shown. Then, we use these candidate pairs as the test function pair, and put them into the KNN classifier to get the potential vulnerability types set. At this time, there are only eight pairs left including the four vulnerable function pairs: sub_40124e:sub_40101c, sub_400f80:sub_4010e2, sub_4012a0:sub_401175, sub_400f2e:sub_400e57. Then through the intersection of each pair, we get a more accurate result.

U.Fun	P.Fun	C.Type	T.Type
sub_40124e	sub_40101c	Type_Confusion	Type_Confusion
sub_400f80	sub_4010e2	Type_Confusion	Type_Confusion
sub_4012a0	sub_401175	Type_Confusion	Type_Confusion
sub_400f2e	sub_400e57	Type_Confusion	Type_Confusion
· · ·	· · ·	· · ·	· · ·

TABLE III: The Final Result of CWE843

[1] U.Fun: Unpatch function, P.Fun: Patch function, C.Type: Candidate Type, T.Type: True Type

V. Conclusion And Future Work

We presented a system to efficiently identify already known vulnerability with the patch file information in binary code across two operating systems. In the preparation phase, two code bases of known vulnerability functions and corresponding patched functions are analyzed and their numeric features and SimHash are stored. When a new vulnerability and its corresponding patch are published, we always can't know the detail information about them. Our approach employs a three-stage method to quickly identify the vulnerability type of the program and which function is the buggy function. The first stage relies on the binary comparison technique to get the function-pairs which may be the vulnerability function and corresponding patch function. Then at the second stage, we extract the digital features and SimHash to retrieve very similar functions based on the KNN algorithm. These functions serve as candidates to the next stage. In the end, we make an intersection of the two candidates set to get the potential vulnerability types. Overall, our method can reduce the false positive and the cost of time doesn't increase too much.And we implemented our methods in a tool call SemHunt and evaluated its efficacy on several program and their patch files with CWE vulnerabilities.

Acknowledgment

We would like to thank Yong Tang for detailed discussions and suggestions about the algorithms used in this paper, and the reviewers from our laboratory for providing useful comments on this paper. This work is partially supported by the National Science Foundation(NSF) China 61472437.

References

[1] J. Ming, M. Pan, and D. Gao, *iBinHunt: Binary Hunting with Inter-procedural Control Flow*. Springer Berlin Heidelberg, 2012.

[2] Z. Wang, K. Pierce, and S. Mcfarling, "Bmat – a binary matching tool for stale profile propagation," vol. 2, p. 2000, 2002.

[3] D. Gao, M. K. Reiter, and D. Song, "Binhunt: Automatically finding semantic differences in binary programs," in *Information and Communications Security, International Conference, ICICS 2008, Birmingham, Uk, October 20-22, 2008, Proceedings*, pp. 238–255, 2008.

[4] K. Riesen, M. Neuhaus, and H. Bunke, *Bipartite Graph Matching for Computing the Edit Distance of Graphs*. Springer Berlin Heidelberg, 2007.

[5] H. Flake, "Structural comparison of executable objects," in *IEEE Conference on Detection of Intrusions and Malware Vulnerability Assessment*, pp. 161–173, 2004.

[6] T. Dullien and R. Rolles, "Graph-based comparison of executable objects (english version)," 2005.

[7] M. Bourquin, A. King, and E. Robbins, "Binslayer: accurate comparison of binary executables," 2013.

[8] L. Liu, B. S. Wang, Y. U. Bo, and Q. X. Zhong, "Automatic malware classification and new malware detection using machine learning*,"

[9] J. Feist, L. Mounier, and M. L. Potet, "Statically detecting use after free on binary code," *Journal of Computer Virology and Hacking Techniques*, vol. 10, no. 3, pp. 211–217, 2014.

[10] S. Eschweiler, K. Yakdan, and E. Gerhards-Padilla, "discovre: Efficient cross-architecture identification of bugs in binary code," in *The Network and Distributed System Security Symposium*, 2016.

[11] Y. Shoshitaishvili, R. Wang, C. Salls, N. Stephens, M. Polino, A. Dutcher, J. Grosen, S. Feng, C. Hauser, C. Kruegel, and G. Vigna, "Sok: (state of) the art of war: Offensive techniques in binary analysis," 2016.

[12] 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," 2016.

[13] Y. Shoshitaishvili, R. Wang, C. Hauser, C. Kruegel, and G. Vigna, "Firmalice - automatic detection of authentication bypass vulnerabilities in binary firmware," 2015.

A Tailor made System for providing Personalized Services

Mario Casillo
DII
University of Naples "Federico II"
Naples, Italy
{mario.casillo}@unina.it

Francesco Colace, Francesco Pascale
DIIn
University of Salerno
Fisciano (SA), Italy
{fcolace, fpascale}@unisa.it

Saverio Lemma, Marco Lombardi
SIMASLab
University of Salerno
Fisciano (SA), Italy
{slemma, malombardi}@unisa.it

Abstract—Today, the existing technologies, such as smartphone and other pervasive devices, can be used for context awareness and help people to undertake conscious choices. In fact, unlike what happened in the past, all of these data are managed and contextualized for the particular application. We need to understand, select, treat and finally opportunely expose these data. In this scenario, there are many Context Aware applications.

This paper introduces a tailor made system for providing personalized services and it is based on a graphical formalism for the context representation: the Context Dimension Tree.

This system provides the needed information about places that are of great interest for the visitors, selecting them using user preferences. A case study is applied to an event in Salerno, an Italian town, called Artist's Lights. Finally, an experimental campaign has been conducted, obtaining interesting results.

Keywords—Context-Aware Computing, E-Citizenship, Adaptive Systems.

I. INTRODUCTION

The Italian towns have a cultural heritage that often do not succeed in being completely enhanced. The natural, artistic and cultural resources present in the Italian towns, above all the smallest ones, many times remain hidden and are not enjoyed by the tourists[12]. This problem typology becomes even more important when the tourist has few hours to visit a town: think, for instance, about some passengers of a cruise who in few hours have to visit an unknown place. The problem arises also for those people who, for work, live an experience in a town that they can visit in little time. Where to eat? What to see? How to move? These are the typical questions that such a user makes when he/she is in a station, an airport or a harbor. If in the big towns there are pre-constituted itineraries that can be easily made by the tourists, this is not always true in towns of little or medium dimension that, even if they have a cultural heritage absolutely interesting, often risk of not enhancing it completely.

On second thoughts, information necessary for the enhancement of the resources of a town are, in many cases, already present on the web: the social networks have much information about the resources present in a town. On the other hand, also the public institutions, usually, develop some contents in support of the cultural resources present in the territory, but not present in places not easily reachable by the tourists, above all the foreign ones. Moreover, often, there are also services that can be useful for a tourist who unlikely knows to find them. Therefore, it is necessary to create a framework that can integrate contents and services to support a user inside a certain territorial context [11].

When the main aim is to create applications, devices, and systems that are easy to use, it is essential to understand the context of use. Today, with context-aware computing, it can be possible to consider the situation of use not only in the design process, but in real time while the device is in use. Generally in Human-Computer Interaction (HCI) [8] the main aim is to understand the user and the context of use and create designs that support the major anticipated use cases and user use situations [9]. In Context-Aware Computing on the other hand, making use of context causes a fundamental change: it can supports more than one context of use that are equally optimal. At runtime – when the user interacts with the application — the system can decide what is the current context of use and provide a user interface specifically optimized for this context. With context awareness, the job of designing the user interface typically becomes more complex as the number of situations and contexts which the system will be used in usually increases. In contrast to traditional systems, they are not designed for a single -or a limited set - of contexts of use, but for several contexts. The advantage of this approach is that it can provides optimized user interfaces for a range of contexts [10].

One of the most important part is select and describe all user situation: user's actions taken with his/her mobile device commonly involve directly or indirectly more than the device holder, thus resulting the usage context to have impact on more than just one person. This information are very important in order to understand what actions to take by context awareness applications and provide pervasive and sometimes indispensable indications for the user.

In this paper is described a context-aware approach for promoting tourism events in Salerno, a town in South of Italy. This work will be organized in this way: in the following paragraph, related works are analyzed, especially for the Mobile Context Awareness. In this paper, the purpose is to give an answer to the problem of the context representation using the Context Dimension Tree (CDT) formalism [21]. Then, it will be present a Context Aware

DOI reference number: 10.18293/SEKE2017-151

App able to promoting tourism events. Some experimental results will be described in the last part of the paper.

II. RELATED WORKS

Today there are many works in research that focus the attention on Context Awareness, particularly in the mobile applications. As seen previously, an app lends itself well to this type of persasive solutions, because today every person has a smartphone connected to the network and it is possible, using the right tools, get all those useful contextual information that can be used to provide an appropriate recommendation to the user. In this context, the main effort has been to retrieve the contextual data of the event or situation of interest and then treat them adequately in order to provide pervasive services to the users.

In [16], a solution has been proposed: an android widget that recommends apps, that through calls and SMS of user, depending on the day/hour of the week and also based on the past behavioral pattern of user activity. The recommendations can be extremely helpful for the user that finds information quicker and effortless.

In [17] a context-aware mobile language learning application is presented, that it focused on providing language support to users living in foreign countries. It suggests context-relevant vocabulary, considering usage context attributes such as user's gender, geolocation and native language. The application architecture is composed by three components which are able to dynamically acquire and analyze the usage context to provide the corresponding information based on knowledge domain. In this work is introduced the Contextual Language Learning (CoLaLe) application, which was designed to support foreign language learning for overseas students. In [2] a novel method to estimate daylight illumination has been proposed using this information in outdoor augmented reality (AR) applications to render virtual objects with coherent shadows. The illumination parameters are acquired in real time from context-aware live sensor data. The particularity is that the Sun's position is calculated based on the user location and time of day, with the relative rotational differences estimated from a gyroscope, compass and accelerometer.

In [20] a novel solution to overcome problems caused by user movement in the mobile cloud computing context is proposed. To this end, a mobility model and a fault tolerance mechanism for optimizing the offloading decision are introduced. Simulation results showed that this proposal can reduce the execution time of mobile applications. This solution considered different parameters that influence offloading like bandwidth, amount of transmitted data and user mobility, with extension to existence prediction models with the aid of Markov chain to find the optimal solution.

In [19] ClickSmart is proposed: a viewpoint recommendation system that can assist a user in capturing high-quality photographs at well-known tourist locations. ClickSmart can provide real-time viewpoint recommendation based on the preview on the user's camera, current time, and user's geolocation. It makes use of publicly available geotagged images along with the associated metadata for learning a recommendation model.

In [15] a context-aware application is proposed and it provides different user authentication methods that are set up according to the auto-detection of areas designated as safe zones by the user. This application aims to improve the overall security of the content of a given device by securing individual applications.

In [24] the context-aware recommendation approach has been introduced and it integrates network location context, user context and demand context. An extensible meta-model is proposed in this system, thus the description of class property can be scalable. The consistent representation of service and user demand will provide convenience for service matching and service filtering. In order to ensure real-time update of service information, this work puts forward three methods: spontaneous service updating, delayed service updating based on accessing, as well as delayed service updating periodically.

In [3] a novel technique for finger tap detection using the built-in speakers as primary sensors is introduced, named TouchSpeaker. Speakers used as sensors (i.e., as microphones) can overcome the aforementioned limitations; these are passive devices, consuming no power for sensing (except for ADconversion).

In [23] a novel approach to recommend services on mobile devices to user has been proposed. This system is composed by three modules: a user behavior model by taking advantage of user's mobile context information like time and location to describe the user states; a model to explain how the sequential service invocations are generated by analyzing the collected sequential history record of mobile users; a logistic model tree approach to determine user state according to given mobile context information, and recommend services to user according to his user state.

In [22] has been presented context modeling and management approaches intended for the optimization of smart interactions and services, discusses the main challenges and requirements of context-awareness in the smart Internet, and provides a feature-based framework useful for the evaluation and implementation of context modeling and management mechanisms.

In [14] has been presented a Semantic Engine of the AMBIT (Algorithms and Models for Building context-dependent Information delivery Tools) architecture in order to focus on the knowledge management foundations that are using to laying this system.

In the next paragraph, an approach to contextualization contents and services using the Context Dimension Tree will be presented.

III. CONTENTS AND SERVICES CONTEXTUALIZATION

For contextualization is meant the act or process of putting information into context; making sense of information from the situation or location in which the information was found. This means that to have the appropriete requests for a context-aware system, it needs having a right model. In order to make contextualized queries, it is necessary to define a model for the representation and management of the context itself, which allows filtering the resources obtained, on the basis of contextual parameters (user position, user profile, user friends, etc.): this operations are made through CDT.

Therefore, the result shows itself like a well-organized information that presents a general introduction about the place reached by the user, according to his/her interests and enriched with the experiences shared by similar users, and a

list of the main suggested attractions about the near places visited by the friends.

In particular, CDT is used to be able to represent, in a graphic form, all possible contexts that you may have within an application. CDT plays a fundamental role in tailoring the information space according to the user's information needs, as well as an analysis of relevant features of context models. It is thus important to notice that this notion of context is strictly connected to the considered application and is not meant to model the general knowledge concerning one or more areas of interest, a situation where a data schema, or a domain-ontology may be better suited [4, 6, 7].

CDT is a tree composed of a triad <r; N; A> where r indicates its root, N is the set of nodes of which it is made of and A is the set of arcs joining these nodes.

A dimension node, which is graphically represented by the color black, is a node that describes a possible dimension of the application domain; a concept node, on the other hand, is depicted by the color white and represents one of the possible values that a dimension may assume. Each node is identified through its type and a label.

The children of the root node r are all dimension nodes, they are called top dimension and for each of them there may be a sub-tree. Leaf nodes, instead, must be concept nodes. A dimension node can have, as children, only concept nodes and, similarly, a concept node can have, as children, only dimension nodes. In addition to nodes, you can use other elements: the parameters, which may arise both from a dimension node (graphically represented by a white square) and from a concept node (white triangle), submitting them to particular constraints. In fact, a concept node can have more than one parameter, while a dimension node can have only a parameter and only in case it has not already children nodes. The introduction of parameters is due to their usefulness in shaping the characteristics that can have an infinite or very high number of attributes. For example, a node representing Cost dimension risks having a high number of values that should be specified by as many concept children nodes. In a similar case, it is therefore preferred to use only one parameter, whose value will be specified in each case. Leaf nodes, in addition to concept nodes, can also be parameters. In general, each node has a parameter corresponding to a domain, dom(nP). For parameter nodes connected to concept nodes, the domain can be a set of key values from a relational database, while in case of parameter nodes connected to dimension nodes, the domain is a set of possible concept nodes of dimension.

Therefore, CDT is used to systematically describe the user needs, and to capture the context the user is acting in. It plays a fundamental role in tailoring the target application data according to the user information needs. Once the possible contexts have been designed, each must be connected with the corresponding view definition. In this ways, when the context becomes current, the view is computed and delivered to the user [18].

In fact, through the CDT, it is possible, after analyzing the domain of application, to express the size characteristics and values they can take in a graphical way by, respectively, dimension nodes and concept nodes or parameters.

The assignment to a dimension of one of its possible values is a context element. The context element can be considered the main feature of the application, by which a context can be decomposed. The moment you make the formulation of the context, you must specify all the context elements that are part of it and that enable its creation. Any context is expressible by an "and" combination of the context elements to which they are peculiar.

By definition, you can begin to understand how you will create views based on data relating to each context; in fact, they will be built starting from the portions of the database and then from the partial views, associated to the context element that takes part into context information.

The CDT elaboration is composed of methodologies and phases to obtain contextual resources. The methodology has been realized in order to manage the database and to carry out reductions of their content based on the context. The purpose is to help the designer in the definition of all contexts relevant to the considered application and, later, in the association to each context of the portion of the database containing the relevant data about the context.

The methodology consists of three main phases, which we will see in detail later: design phase of the CDT, definition phase of partial views and composition phase of global views [1].

1. Design phase of the Context Tree: in this phase, the CDT is designed to identify significant context elements for the considered application. In fact, it focuses on the definition of contexts and on the elements that compose them. These contexts must be identified and shaped, indicating particular elements that characterize each of them. As it has been said, it is available a special tool called CDT to make context design. Various CDT were made for specific environments in order to represent and manage a multitude of different contexts and in order to identify, represent, preserve and make available cultural points for each type of user.
2. Definition phase of partial views: after the definition of all the contexts and their context elements, in this step a different portion of the database is associated to each context element, containing the relevant data for it. In practice, the goal is to find the appropriate value for a given dimension, in order to obtain, by means of the values of all the dimensions, a valid query and specific to the context in which the user is located.
3. Composition phase of global views: this is the phase where you have the automatic generation of views associated with each context, which is made starting from partial views associated with context elements. After the creation of the global views of the contexts, the answers to questions that will be asked to the system will be developed from these views and, in particular, from the view associated with the context in which you are located when the query is performed.

In particular, once defined the values for each dimension, you can use all the information obtained in order to identify the right context and offer contextual resources for the user.

In figure 1, it is shown a general designed CDT, called Meta CDT, which is the starting point for the design of a specific CDT that can be exploited in contextual applications [13]. You may note six top dimensions, which correspond to the questions of the 5W1H method: Location (WHERE), Role (WHO), Time (WHEN), Situation (HOW), Interest (WHAT) and Utilization (WHY).

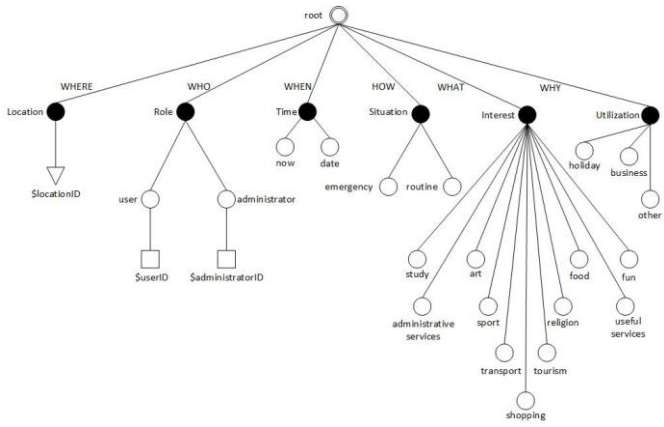


Figure 1. Meta CDT.

In particular, there are two types of users and eleven categories of interests. In this case, as shown in figure 2, a partial view could be related to dimension "Role": once logged in, the application is able to recognize the user and to know more precisely whether he/she is, for example in tourist areas, a resident or a tourist. Thus, the value "tourist" of dimension "Role" is a partial view for the current context: using this knowledge, you can exclude certain resources, not suitable or useful to the tourist role.

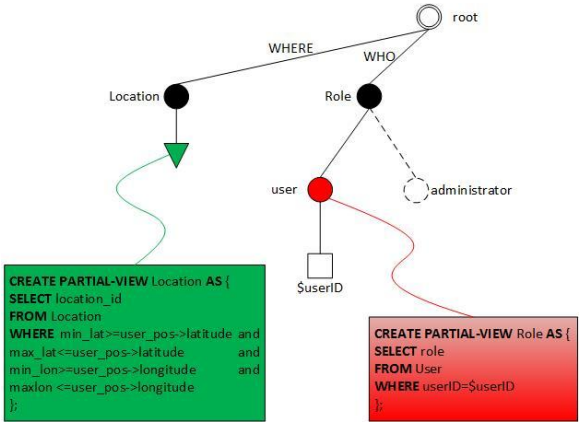


Figure 2. Example of Partial Views.

A context element is defined as an assignment d_name_i = value, where d_namei indicates a possible size or undersize of CDT (it is the label of a dimension node), while value may represent the label of one of the concept nodes that are children of the considered dimension node or the value of a parameter referring to one of these concept nodes or the value of a parameter referring to the considered dimension node.

For example, these assignments are possible context elements:

Interest = "tourism", Location = $locationID (for example, ID = 3), Role = $userID->role (for example, ID = 15), Utilization = "holiday".

A context is specified as: $\wedge(d\_name_i$ = value): it is defined as an "and" among different context elements.

Several context elements, combined with each other by means of an "and", damage, therefore, the origin of a context [5]

For example, a possible framework that can be obtained from the previously seen CDT, through the context elements that we have listed, is:

C = (Location = $locationID (ID=3)) ∧ (Role= $userID->role (ID=15))
∧ (Time = "now") ∧ (Situation = "routine")
∧ (Interest = "tourism") ∧ (Utilization = "holiday")

The context is defined as a user, interested in tourism, who uses the contextual app on vacation, in a called place.

IV. A CONTEXT AWARE APP FOR PROMOTING TOURISM EVENTS

In this section, we will present a contextual app designed and implemented according to what was described previously. In particular, we have thought to apply the approach in the context of Artist's Lights Christmas event that every year is held in Salerno (Regione Campania in Italy) and that involves hundreds of thousands of tourists. From November to January, with light installations, some by local artists exclusively for Salerno, scattered through the main streets and in the most beautiful and attractive corners of the city center.

In this phase, we have collected the services and contents potentially useful for the citizens and situate them on the map defining the activation zones (figure 3).

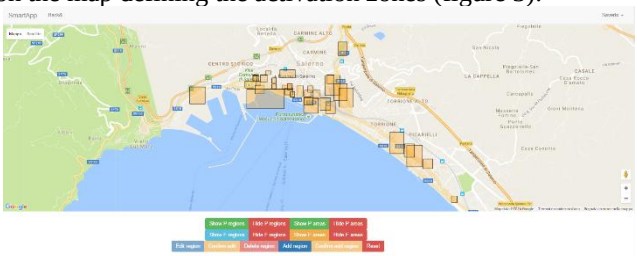


Figure 3. Definition of the activation areas of services and contents.

Moreover, we have defined the different typologies of citizens (tourist and expert user) associating them to a previously established set of services and contents. Having the town a series of Christmas contents, we have developed services and contents in support of them too.

All information about places of worship and shops has been uploaded, for any building or area of potential.

The App has been developed with hybrid technologies (Ionic Framework and Apache Cordova) to allow an easier publication both in Android and Apple environment (figure 4).

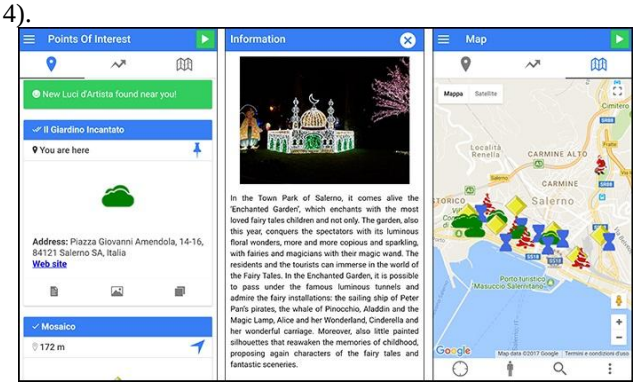


Figure 4. Screenshots with some features of contextual application.

The experimental phase aims to evaluate the proposed contextual model. Initially, the App has been presented to the population in November 2016. They have been involved overall about 2000 tourists between 18 and 60 years old.

During this event, the app has been installed on the mobile devices of the tourists.

After having interacted for some days with the application, the participants have then answered on the basis of the Likert scale to fourteen statements, divided into four sections. To every question present in the section, five possible answers have been associated: I strongly agree – I agree – Undecided (Neither agree nor disagree) – I disagree- I strongly disagree.

The questionnaire in detail is the following:

Section A: App – Context

- A1. The App gives the user tailor-made contents and services in the right place.
- A2. The App allows the user to know several item of the Artist's Lights.
- A3. The App supplies services according to the interests selected in the user profile.

Section B: App – Further aspects

- B1. Information about each item of Artist's Lights is very useful.
- B2. The contents, such as descriptions and images, are of high quality and represent one of the strong points of Artist's Lights
- B3. The services associated to the items allow a higher immediacy than a classic research on the Internet.

Section C: App – Functionality

- C1. The plan itinerary service allows easily realizing an itinerary in the Artist's Lights according to the user's preferences.
- C2. The explore surroundings service is very useful to know what there is nearby and eventually reach them.
- C3. The functionality of QR code in inner environments can be well used.

Section D: App – Future developments

- D1. It would be interesting to have a higher integration with the main social networks.
- D2. It would be interesting to insert the available time in the plan itinerary service.

Table 1 presents a synthesis of the answers of the participants to each declaration.

TABLE I: Experimental results

Likert Scale	Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree
A1	698	996	186	69	51
A2	941	723	187	89	60
A3	732	966	207	54	41
B1	698	974	148	102	78
B2	798	845	234	78	45
B3	784	969	196	33	18
C1	1012	771	124	44	49
C2	824	942	112	69	53
C3	781	899	199	74	47
D1	835	818	196	87	64
D2	829	745	287	87	52

As shown in this table, of the 2000 participants who have interacted with the application, many agree and/or strongly agree that the system gives appropriate contextual information about the place, further aspects and functionality are very useful and future developments are interesting. Instead, only in few cases, the participants do not are particularly satisfied.

As can noticed from the figure 5, users show great appreciation for the app. In general, they appreciated the proposed contents and services.

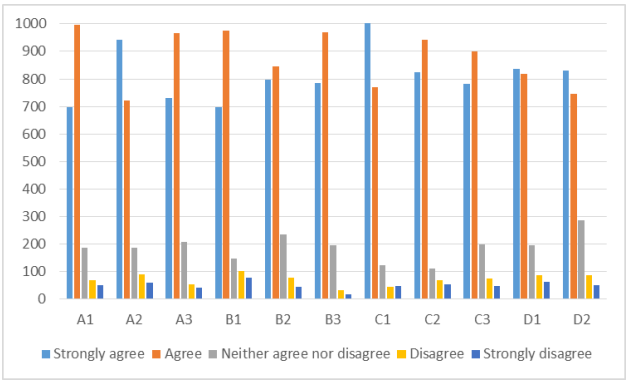


Figure 5. Graphic analysis of experimental results.

V. CONCLUSIONS

Based on the concept of Context Dimention Tree, in this paper is presented the use of Context-Aware approach in order to allow contextual decisions for users according to his/her needs. CDT is a graphical formalism able to model a context by the approach of the 5W1H method.

To obtain this results, a mobile application was developed keeping track of users needs, based on context, providing personalized services. The App bases its 'contextual' functioning on the adoption of the CDT that is able to shape the context and the actions to implement.

The App has been designed for the Christmas event, Artist's Lights, which was held in Salerno, a city in the south of Italy. The following activities have as purpose the application of the proposed methodology to more complex environments, for dimension and number of potential places to manage.

REFERENCES

[1] G. Annunziata, F. Colace, M. De Santo, S. Lemma, and M. Lombardi, "ApPoggiomarino: A Context Aware App for e-Citizenship," The 18th International Conference on Enterprise Information Systems (ICEIS), 2016.

[2] J. Barreira , M. Bessa, L. Barbosa and L. Magalhaes, "A Context-Aware Method for Authentically Simulating Outdoors Shadows for Mobile Augmented Reality, "IEEE Transactions on Visualization and Computer Graphics, Volume: PP, Issue: 99 , 2017.

[3] J. Beysens, A. Chiumento, S. Pollin and M. Li, "TouchSpeaker, a Multi-Sensor Context-Aware Application for Mobile Devices," IEEE International Workshop on Signal Processing Systems, 2016.

[4] C. Bolchini, C. Curino, E. Quintarelli, F.A. Schreiber, and L. Tanca, "A survey of context models and a preliminary methodology for context driven data view definition," Technical Report 4.1, ESTEEM Project, 2006.

[5] M. Casillo, L. Cerullo, F. Colace, S. Lemma, M. Lombardi, and A. Pietrosanto, "An Adaptive Context Aware App for the Tourism," In Proceedings of the The 3rd Multidisciplinary International Social Networks Conference on SocialInformatics 2016, Data Science 2016 (p. 26). ACM, 2016.

[6] M. Casillo, F. Colace, S. Lemma, M. Lombardi, and A. Pietrosanto, "An Ontological Approach to Digital Storytelling," In Proceedings of the The 3rd Multidisciplinary International Social Networks Conference on SocialInformatics 2016, Data Science 2016 (p. 27). ACM, 2016.

[7] F. Colace, M. De Santo, L. Greco, and P. Napoletano, "Weighted Word Pairs for query expansion," Information Processing and Management, 51 (1), pp. 179-193, 2015.

[8] F. Colace, P. Foggia, and G. Percannella, “A probabilistic framework for TV-news stories detection and classification,” IEEE International Conference on Multimedia and Expo, ICME 2005, 2005, art. no. 1521680, pp. 1350-1353, 2005.

[9] F. Colace, M. De Santo, and L. Greco, “An adaptive product configurator based on slow intelligence approach,” International Journal of Metadata, Semantic and Ontologies (IJMSO), Vol.9, No.2, pp. 128-137, 2014.

[10] F. Colace, L. Greco, S. Lemma, M. Lombardi, D. Yung, and S.K. Chang, “An Adaptive Contextual Recommender System: a Slow Intelligence Perspective,” The Twenty-Seventh International Conference on Software Engineering and Knowledge Engineering (SEKE), pp. 64-71, 2015.

[11] F. Colace, M. De Santo, L. Greco, V. Moscato, and A. Picariello, “A collaborative user-centered framework for recommending items in Online Social Networks,” Computers in Human Behavior 51: pp. 694-704, 2015.

[12] F. Colace, M. De Santo, S. Lemma, M. Lombardi, A. Rossi, A. Santoriello, A. Terribile, and M. Vigorito, “How to Describe Cultural Heritage Resources in the Web 2.0 Era?,” Proceedings - 11th International Conference on Signal-Image Technology and Internet-Based Systems (SITIS), art. no. 7400656, pp. 809-815, 2015.

[13] F. Colace, L. Greco, S. Lemma, M. Lombardi, F. Amato, V. Moscato, and A. Picariello, “Contextual Aware Computing and Tourism: A Case Study,” The Eleventh International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), pp. 804-808, 2015.

[14] R. Martoglia, “AMBIT: Semantic Engine Foundations for Knowledge Management in Context-dependent Applications,” Proc. of the 27th International Conference on Software Engineering and Knowledge Engineering (SEKE), 2015.

[15] M. Masango, F. Mouton, A. Nottingham and J. Mtsweni, “Context Aware Mobile Application for Mobile Devices,” Information Security for South Africa (ISSA), 2016.

[16] N. Mendes, B. Alves and S. Paiva, “Context-aware mobile recommender system based on activity patterns,” 17th IEEE International Conference on Mobile Data Management, 2016.

[17] R. Morales, B. Igler, S. Bohm and P. Chitchaipoka, “Context-Aware Mobile Language Learning,” 12th International Conference on Mobile Systems and Pervasive Computing, MobiSPC, 2015.

[18] C. Parent, K. Schewe, V. Storey, and B. Thalheim, Conceptual Modeling - ER 2007, 26th International Conference on Conceptual Modeling, Auckland, New Zealand, November 5-9, 2007.

[19] Y. S. Rawat and M. S. Kankanhalli, “ClickSmart: A Context-Aware Viewpoint Recommendation System for Mobile Photography,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 27, no. 1, 2017.

[20] R. Roostaei and Z. Movahedi, “Mobility and Context-Aware Offloading in Mobile Cloud Computing,” IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress, 2016.

[21] L. Tanca, C. Bolchini, C. Curino, and F.A. Schreiber, “Context integration for mobile data tailoring,” Italian Symposium on Database Systems (SEBD), pp. 48-55, 2006.

[22] N. M. Villegas and H. A. Mller, “Managing dynamic context to optimize smart interactions and services,” The Smart Internet, volume 6400 of Lecture Notes in Computer Science, pages 289– 318. Springer Berlin Heidelberg, 2010.

[23] Z. Xiang, S. Deng, S. Liu, B. Cao and J. Yin, “CAMER: A Context-Aware Mobile Service Recommendation System,” IEEE International Conference on Web Services, 2016.

[24] Z. Zhao, H. Chen, R. Li and Z. Wang, “An Approach to Context-aware Service Pushing for Mobile Computing,” IEEE International Conference on Mobile Services, 2016.

Fault Interference and Coupling Effect

Chunrong Fang, Yang Feng, Qingkai Shi, Zicong Liu, Shuying Li, Baowen Xu*
State Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210093, China
E-mail: *bwxu@nju.edu.cn

Abstract

Any program may contain more than one fault, and these faults may interfere with each other in a variety of ways. Software behavior may be affected by the interference, resulting in some uncertain results. Such results have negative impact on many software engineering tasks, including regression testing, fault localization, debugging, fault clustering etc. Therefore, understanding the interference becomes an important topic. This paper investigates the fault interference from the perspective of software construction. We introduce the coupling of software construction in order to explain the reasons for fault interference. We observed that different types of coupling may cause three kinds of fault interference and have different probabilities to make the software strike the fault interference traps. We conducted a preliminary experiment on four industrial programs. The results show that our approach gives a good explanation on fault interference.

1 Introduction

Software is in the process of an explosive growth on complexity. Unfortunately, the maintenance process grows more complicated. This brings more difficulties as the number of faults increases and multiple faults usually interfere with each other in real-world software. Due to multiple bugs interfering with each other, some statistical algorithms for single-bug version of software perform poorly in practice[18, 17]. This problem becomes even more difficult as the number of faults in software increases, and real-world software always have multiple faults. Existing statistical algorithms that focus solely on identifying predictors that correlate with program failure perform poorly, especially when there are multiple bugs. To alleviate the problem, failed test cases are segregated into clusters, such that the failed test cases in each cluster can be mapped to the same causative fault in [8, 15]. However, such approach is impractical. Multiple faults may interfere with each other in complex ways, resulting in an inaccurate clustering results. This may lead to an critical and negative impact on some essential software engineering tasks, such as regression testing, fault localization, fault clustering, etc.

However, only a few studies have been conducted to investigate the root cause of faults and their interferences. Debroy and Wong [2] explored the idea of fault interference (FI) by examining the Siemens suite. It was observed that FI is a common phenomenon in software, whereas the reasons why FI exists and how to alleviate FI are not given. Nicholas and James [5] conducted an experiment on some programs of a larger scale, and they got a persuasive result and presented a more thorough explanation on such phenomena.

In this paper, we performed a comprehensive experiment on four industrial C programs. We gain insights in how faults interfere with each other and some potential patterns are mined. This can help developers understand how coupling affects the interaction between multiple faults, which helps them improve their design and implementation. Further, distances between faults are defined to predict patterns that faults interaction will occur and simplify the process to locate faults. Moreover, our work may help software developers in at least three aspects: understanding and improving software tasks mentioned above, obtaining a new perspective to understand software design and software testing, along with learning how to handle and avoid fault interference.

1.1 Motivation

Software coupling is the degree to which each program module relies on each one another[16]. High coupling degree will expose some disadvantages to the software system: needing a ripple effect to changes in other modules, such that it may be hard to reuse and test, lying confusions in the program logic. Absolutely, FI results from coupling, though not all of coupling will generate FIs.

In order to understand the coupling problem, this work examines one aspect of fault behavior: fault interference,

DOI reference number: 10.18293/SEKE2017-077

which has been observed in a variety of research studies. FI is defined as the change in the behavior of one or more faults due to multiple faults on the FI working together. This work provides a deeper investigation of the FI phenomenon. We expect that a thorough understanding of the root causes of FI will enable research areas that address issues with faults to better cope with the challenges that arise from the presence of multiple faults.

1.2 Contribution

In this paper, we make the following contributions:

- We introduce software engineering knowledge to investigate fault interference in multiple fault programs. We define the fault interference problem. As far as we know, this is the first study to introduce engineering perspective on fault interference with multiple faults.
- We introduce software coupling to explain the root causes for different types of fault interference, based on several types of software coupling. No pioneering studies have made such attempts.
- We conducted a comprehensive experiment on industrial subject programs, and the results show that coupling frequency and coupling values contribute to fault interference.

The rest of paper is organized as follows. Section 2 explains the detail of our approach. The experimental design and the result analysis are presented in Section 3. Related work and discussion are presented in Section 4. The conclusion and future work are presented in Section 5.

2 Approach

2.1 Fault Coupling

Some empirical studies [1, 4] show that the designs and implementations with low coupling and high cohesion will lead to more reliable and maintainable products. The industry and academia have been proposed some theories to measure the coupling. Experts have reached an agreement on the coupling type and hold clear definition and threats in software construction of them [11]. We introduce coupling to measure and predicate fault interference. The following five types [] of coupling are observed in our investigation of fault interference. All of them lead to one or more types of fault interference.

We list them in the coupling degree ascending order:

- C1: Data Coupling, communication via scalar parameters;
- C2: Stamp Coupling, dependency induced by the type of structured parameters;
- C3: Control Coupling, parameters are used to control the behavior of a module;
- C4: Common Coupling, communication via shared global data;
- C5: Content Coupling, one module shares and/or changes the definition of another nodule.

2.2 Fault Interference

There exist some studies on fault interference on programs with multiple faults. However, to the best of our knowledge, the formal definition on fault interference is absence. In this section, we propose the definition of fault interference and then category it into three types.

Definition 2.1 (Fault Interference) *Given a test case t, a program P and some faults $f_1, \cdots, f_k$, it is called that $f_1, \cdots, f_k$ interfere on t if*

$$O(P_{f_1+\cdots+f_k}, t) \neq O(P_{f_1}, t) \vee \cdots \vee O(P_{f_k}, t) \qquad (1)$$

$O(P, t)$ is a boolean function of oracle. That is, $O(P, t) = 1$ indicates that P is failed by running t, otherwise, $O(P, t) = 0$. P_{f_i} is a program with one single fault f_i and $P_{f_1+\cdots+f_k}$ is a program with multiple faults $f_1, \cdots, f_k$. Equation (1) implies an inconsistent behaviors between all single versions and a multiple version. To give a deep insight of the fault interference, we introduce the following definition.

Definition 2.2 *Given a program P and some faults $f_1, \cdots, f_k$ interfere on t, (1) the interference is called β-I if $O(P_{f_1+\cdots+f_k}, t) = 1$;(2) the interference is called α-I if $O(P_{f_1+\cdots+f_k}, t) = 0$; (3) For α-I in (2), the interference is called α_1-I if $\forall i \in [1, k]$, $O(P_{f_i}) = 1$, α_1-I if $\exists i \in [1, k]$, $O(P_{f_i}) = 0$.*

For fault interference, as $O(P_{f_1+\cdots+f_k}, t) \neq O(P_{f_1}, t) \vee \cdots \vee O(P_{f_k}, t)$,$O(P_{f_1+\cdots+f_k}, t) = 0$ implies that $O(P_{f_i}, t) = 1$ for some i; $O(P_{f_1+\cdots+f_k}, t) = 1$ implies $O(P_{f_i}, t) = 0$ for all i. α-I indicates a devoid phenomenon of fault interference. β-I indicates an enhancement phenomenon of fault interference. In particular, we further classify α-I into two types: α_1-I if $O(P_{f_i}, t) = 1$ for all i and α_0-I otherwise. That is, α_1-I could be considered as the strong one of α-I.

Note that there is a key difference between our definition and the previous. As we observed in our experiment, the root causes of α_0-I interference and α_1-I interference are generally different. In our opinion, these two phenomena should not be categorized into one type, though the result seems to be similar. In the rest of this paper, we just use α_0, α_1, β to denote the α_0-I, α_1-I, β-I.

2.3 Research Questions

According to the classical software engineering theory, the higher coupling degree is between two blocks, the more possibly exists unpredictable software behavior. To facilitate the discussion, we use $S(F)$ to denote the faults set of the program, and $B(F)$ the block of fault exists. In this case, we can get the mapping: $S(F) \rightarrow B(F)$. $B(F)$ contains the location information, including line number, fault type, existing function head, involving global parameters. $FI_{\alpha\beta}(FI_i, FI_j) \rightarrow B_i, B_j$ we can get construct information between B_i and B_j, which could present the coupling relationship $C(F_i, F_j)$ between F_i and F_j. In this paper, we will investigate research questions as follows.

- RQ1: How frequently the three types of fault interference happen? This question will investigate in what extent fault interference happens and affects program behaviors. Besides, we are interested in whether all the three types have a same change to show up.

- RQ2: Does more fault coupling counts imply more fault interference? This question aims to find the relation between the number of fault coupling and fault interference.

- RQ3: Does high fault coupling value imply more fault interference? Different from $RQ2$, this questions focuses on whether a higher value coupling means a higher possibility of fault interference.

3 Experiment

3.1 Subject Programs

In order to collect sufficient data about fault interference, we used four subject programs: flex, gzip, space and make, all of which are downloaded along with their test sets from the Software-artifact Infrastructure Repository (SIR)[1]. The four programs have been well studied and used as subject programs in many other essential software engineering exploring experiments. Table 1 presents the detailed information. Thanks to the previous researchers [], we excluded some faults that could not cause any failures independently, which results in reducing the number of faults. Besides, using the same approach in [5], we combine faults by randomly selecting a line or block to replace the excluded ones, ensuring that no faulty block is overlapped and the size of fault set is not changed.

[1]http://sir.unl.edu/portal/index.php

Table 1. Subject Programs

Program	Flex	Gzip	Make	Space
Release No.	2.5	1.1.2	3.76.1	2.0
# LOC	14273	7928	35545	6445
# Functions	162	104	268	136
# Faults	20	16	19	33
# Faulty Versions	190	120	171	528
# Test Cases	567	214	1043	13527

3.2 Experiment Setup

To facilitate manual analysis on the causes of fault interference, we simplify the question: two faults interference situation is the best choice, which is of simplicity and representativeness. In this experiment we gathered the pass/fail status of our four subject programs containing one fault independently as well as two faults at same time.

In order to get the pass/fail status, we compare the outputs of test cases run on the fault-seeded versions with that of the golden version. MD5 algorithm is applied to check whether the outputs are the same or not. If the MD5 value is not equal, the execution was marked as a fail; otherwise, it was considered as a pass. With the help of the pass/fail status matrix of all test cases running on all fault-seeded versions, FI types occurrence times, two bugs interacting with each other were recorded by scripts.

The second step is to analyze the root cause of FI. We gather all the fault information, the fault location(in which LOC), fault type, the function header which the fault statements or parameters belong to, the control flow information the faulty block contains, the global parameters which are involved and the local parameters which are used in the block. Furthermore, Egypt[2] is used to generate the function call graph (FCG) and data flow graph (DFG) to assist the software construction analysis procedure. Finally, we apply the information collected to Formula 2, to get the total and average coupling value (CV) of all types of FI.

3.3 Evaluation

For all subject programs are written in C, as described in [1][4], we introduce the Dhama[4] coupling metric that measures the coupling inherent to an individual module M to evaluate the coupling value when FI occurred. The formula is as following:

$$CV(M) = 1 - \frac{1}{i_1 + q_1 i_2 + u_1 + q_2 u_2 + g_1 + q_3 g_2 + w + r}, \quad (2)$$

where q_1, q_2 and q_3 are weight factor to improve the predicator weight. For data and control flow coupling: i_1 is the

[2]http://www.gson.org/egypt/

Table 2. FI Information

	Flex	Gzip	Make	Space
β FI Occurrence Times	0	229	0	138
α_0 FI Occurrence Times	82	103	15	2552
α_1 FI Occurrence Times	652	79	605	13276
Total FI	734	411	620	15966
Single Bug Execution Times	11340	3424	19817	446391
Two Bug Execution Times	107730	25680	178353	7142256
Total Execution times	119070	29104	198170	7588647
FI Occurrence Percent	0.681%	1.60%	0.34%	0.224%
Average of FI in LOC	0.0514	0.0518	0.0174	2.4772
Average of FI in test case	1.2945	1.9205	0.5944	1.1803
Average of FI in function	4.53	3.95	2.31	117.40

Table 3. Coupling Values of FI

FI		Flex	Gzip	Make	Space
β	Total CV	NA	73.0412	NA	39.6661
	FI Occurrence		229		138
	Average CV		0.3189		0.2874
α_0	Total CV	28.6061	5.5943	3.2166	785.849
	FI Occurrence	82	103	15	2552
	Average CV	0.3488	0.2484	0.2144	0.3079
α_1	Total CV	316.4958	26.5607	176.329	5520.1608
	FI Occurrence	652	79	605	13276
	Average CV	0.4845	0.3362	0.2914	0.4158

input data number, i_2 is the control parameter number, u_1 is the output data number, and u_2 is the output control parameter number. For global coupling: g_1 is the number of global parameters used as data, and g_2 is the number of global parameters used for control. For environment coupling: w is the number of other modules called from module M, and r is the number of modules calling module M; The formula has a minimum value of g_1 and g_2 is the number of global variables used for control. In our experiment, to concentrate on the FI occurrence, we just calculate the faulty block coupling value, and q_1, q_2, q_3 are assigned to 2 as a heuristic estimate. To simplify the experiment, we treat the faulty block as the fault module. When the FI occurred in two faulty block, we just treat the involved parameters as the factor of our evaluation formula.

3.4 Experiment Result and Analysis

The analysis of the experiment result includes three parts: FI occurrence statistic, the cause investigation, and the relation between coupling value and FI analysis, to answer the three questions proposed in II-C. The FI statistic information is presented in the Table 2. The coupling percentage analysis is shown in Fig. 1, where α, β denote the FI types depicted in the previous section, and the coupling value statistic is in the Table 3.

Fig. 1 presents the percent of each coupling effect in each type of FI, about the Fig. 1. We must explain that the occurrence of FI may be not caused by only one type of coupling. To facilitate statistical analysis and assure the precision, we record all of the coupling type when FI occurred. We use CP_T(X) to denote the T type(data, stamp, control, common, content) of Coupling Proportion(CP) which lead to X(α_0, α_1, β) type of FI, and the T type of coupling count in X type of FI is denoted as CCO_T(X). The CP_T(X) calculated as follows.

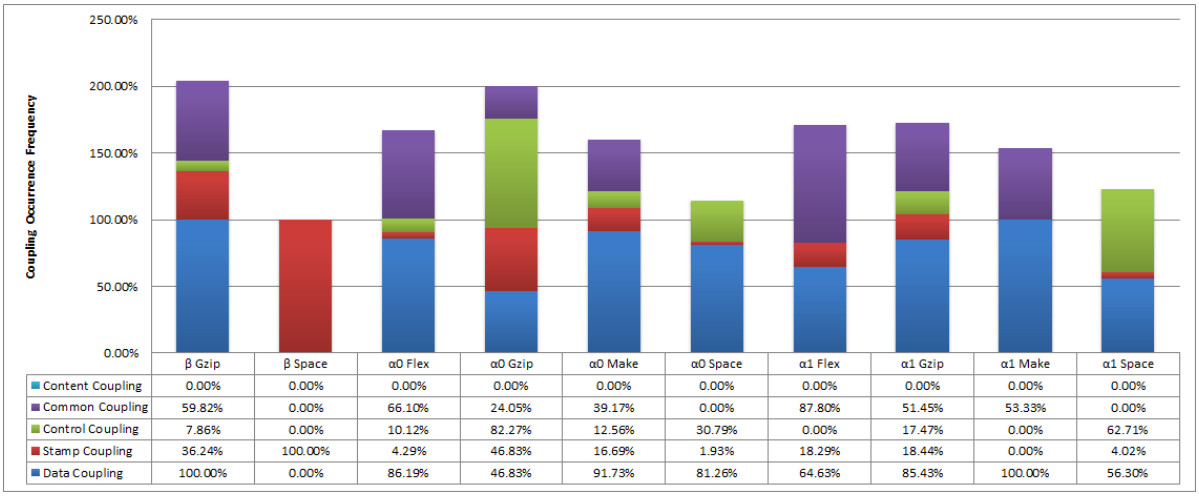
$$CP_T(X) = \frac{CCO_T(X)}{Occurrence\ Number\ of\ X} \qquad (3)$$

In this way, the total coupling proportion leads to one type of FI may be over 100 percent.

RQ1: Frequency of fault interference. It is obvious that, the FI occurred in a low frequency, about 1%. In the experiment, we did not observe the FI type, β, in the flex and make, and only a few (about 0.86% = 138/15966) in the Space. The data of Gzip is totally different from other subject programs. The function of Gzip is to compress or decompress the files. We analyze the aberration by tracing the test cases function call routes on the FCG. And the results show that a very high proportion of the test cases called almost all the functions and that each test case has a high code coverage value. In such case, the test case may have a relative high possibility to cover the faulty block. This is considered as the reason why the α_0 and α_1 FI proportion is relatively low in Gzip.

RQ2:The relation between coupling frequency and fault interference. As shown in the Fig. 1, no content coupling is observed in the programs, which may benefit from the well-defined programming style of the four subject programs. However, based on the analysis on the FI types, it reveals that any kind of coupling could lead to FI, that is, a low data coupling frequency corresponds to a low average FI in test cases and a high common coupling frequency generate a high average FI in test cases.

RQ3:The relation between coupling value and fault interference. According to Table 3, all of the subject programs have a higher average CV in α_1 than that in α_0, which indicates that the occurrence of α_1 always results from a much tighter coupling than that resulting in α_0. When we analyze the source code of α FI involved in faulty blocks, we discovered that the main reasons for α_0 and α_1 are totally different: when the α_0 FI occurring, software usually performs correct behavior, which leads the software to skipping the wrong behaviors. In other words, the program seems to behave correct, and jump over another faulty block. In this way the correct behavior cover the wrong behavior and the program seems correct. Another finding is that about 65% α_0 are caused by the use of global parameters. The most important reason leading to α_1 FI is that



	β Gzip	β Space	α0 Flex	α0 Gzip	α0 Make	α0 Space	α1 Flex	α1 Gzip	α1 Make	α1 Space
Content Coupling	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Common Coupling	59.82%	0.00%	66.10%	24.05%	39.17%	0.00%	87.80%	51.45%	53.33%	0.00%
Control Coupling	7.86%	0.00%	10.12%	82.27%	12.56%	30.79%	0.00%	17.47%	0.00%	62.71%
Stamp Coupling	36.24%	100.00%	4.29%	46.83%	16.69%	1.93%	18.29%	18.44%	0.00%	4.02%
Data Coupling	100.00%	0.00%	86.19%	46.83%	91.73%	81.26%	64.63%	85.43%	100.00%	56.30%

Figure 1. Coupling Percent Analysis

one of the wrong behavior corrects another wrong behavior by coincidence. And such situation almost occurs in the condition branch. The β FI usually occurs in the situation that: the two independent correct behaviors are not strictly or logically correct, the two faulty blocks form a more fatal fault to some software behavior, and the new fault is easier to be detected for the test case.

In our experiment, we just investigate the FI in two faults as the same with Nicholas and James' excellent research[5]. In general, as the program scale and faults number increase, the FI occurrence frequency will increase. The interference of more than two faults may be a more complicated topic.

4 Related Work and Discussion

Fault localization techniques relying on slicing or statistics usually depend on fault behavior. In [12], James et. al. found that multiple faults can introduce noise, which can decline the effectiveness of fault localization tools. Zheng et al. [18] and Liblit et al. [14] also found a similar phenomenon that multiple faults make feature selection difficult. Denmant et al. [3] coverage-based fault location requires an assumption that multiple faults should be independent with each other. Further, DiGiuseppe et. al. [6] found that fault interference has a large impact on the capability of fault localization techniques on multiple faults program. However, these studies focuses on influence of fault interference on fault localization techniques.

Regression Testing[10] aims to ensure new faults are not introduced when software evolve. In [7], a hierarchy of logic faults are investigated on regression testing with sing fault versions. The fault interference may affect the performance of regression testing.[13] found that a fault may not be detected when it is mixed with other faults.

Fault Interference Debroy and Wong[2] explored the idea of fault interference and they examined whether a test suite perform the same on single-fault versions or multiple-fault versions. Their empirical studies suggest that failure masking is a very quite common phenomenon. DiGiuseppe and Jones [5] provided evidence for the prevalence of fault interference and found that faults obscuring is the most prevalent type. Further, they gave a thorough discussion about the adverse effect of fault interference on many existing studies, such as regression testing and fault localization. Comparing to the existing studies, we introduce software coupling and present explanations on the root causes of fault interference.

5 Conclusion & Future Work

Investigating the fault interference root cause is an important task of software engineering. The interference between faults threats many important software engineering tasks. In this paper, we present a theory to explain the fault interference. The main challenge on this topic is that the developers and testers ignore the fault interference; a natural idea is to propose a theory to explain the phenomenon, predict it and avoid it.

We noticed the phenomenon in our experiment and try to apply the framework to analyze it, but we cannot discuss them in detail due to the limited pages. The approach present in our paper, do not cover how to treat the coupling on the popular Object-Oriented programming, and thanks to the smart previous researchers, there exists some mathematical measuring to measure Object-Oriented, Modular, Aspect-Oriented programming. We will introduce these excellent methodology to investigate the FI.

There is little research on revealing and explaining the

fault interference, and the methodology on handling multiple bugs program is in lack. The exploration of multiple bugs is preliminary. There are many aspects about this topic that could be improved or studied in the future. The fault localization algorithm, testing clustering methods, regression testing techniques should take fault interference into consideration. Furthermore, we should build a more detailed theory to explain, predicate and avoid fault interference. For example, predefined specifications with respect faults can be investigated[9]. We will also conduct a more comprehensive experiment to explore the fault interference and take it into consideration to improve other software engineering tasks in the future.

Acknowledgement

This work was partially supported by the National Basic Research Program of China (973 Program 2014CB340702), the National Natural Science Foundation of China (No. 61373013, 61170067), HPI and HPI Research School, Program B for Outstanding PhD Candidate of Nanjing University.

References

[1] J. S. Alghamdi. Measuring software coupling. *Arabian Journal for Science and Engineering*, 33(1):119, 2008.

[2] V. Debroy and W. E. Wong. Insights on fault interference for programs with multiple bugs. In *Software Reliability Engineering, 2009. ISSRE'09. 20th International Symposium on*, pages 165–174. IEEE, 2009.

[3] T. Denmat, M. Ducassé, and O. Ridoux. Data mining and cross-checking of execution traces: a reinterpretation of jones, harrold and stasko test information. In *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, pages 396–399. ACM, 2005.

[4] H. Dhama. Quantitative models of cohesion and coupling in software. *Journal of Systems and Software*, 29(1):65–74, 1995.

[5] N. DiGiuseppe and J. A. Jones. Fault interaction and its repercussions. In *Software Maintenance (ICSM), 2011 27th IEEE International Conference on*, pages 3–12. IEEE, 2011.

[6] N. DiGiuseppe and J. A. Jones. On the influence of multiple faults on coverage-based fault localization. In *Proceedings of the 2011 international symposium on software testing and analysis*, pages 210–220. ACM, 2011.

[7] C. Fang, Z. Chen, and B. Xu. Comparing logic coverage criteria on test case prioritization. *Science China Information Sciences*, 55(12):2826–2840, 2012.

[8] Y. Feng and Z. Chen. Multi-label software behavior learning. In *Software Engineering (ICSE), 2012 34th International Conference on*, pages 1305–1308. IEEE, 2012.

[9] W. Ghardallou, N. Diallo, and A. Mili. Software evolution by correctness enhancement. In *The 28th International Conference on Software Engineering and Knowledge Engineering, (SEKE)*, pages 605–610. KSI Research Inc., 2016.

[10] T. L. Graves, M. J. Harrold, J.-M. Kim, A. Porter, and G. Rothermel. An empirical study of regression test selection techniques. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 10(2):184–208, 2001.

[11] M. Hitz and B. Montazeri. Measuring coupling and cohesion in object-oriented systems. 1995.

[12] J. A. Jones, M. J. Harrold, and J. Stasko. Visualization of test information to assist fault localization. In *Proceedings of the 24th international conference on Software engineering*, pages 467–477. ACM, 2002.

[13] J.-M. Kim, A. Porter, and G. Rothermel. An empirical study of regression test application frequency. In *Proceedings of the 22nd international conference on Software engineering*, pages 126–135. ACM, 2000.

[14] B. Liblit, M. Naik, A. X. Zheng, A. Aiken, and M. I. Jordan. Scalable statistical bug isolation. *ACM SIGPLAN Notices*, 40(6):15–26, 2005.

[15] C. Liu and J. Han. Failure proximity: a fault localization-based approach. In *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 46–56. ACM, 2006.

[16] W. P. Stevens, G. J. Myers, and L. L. Constantine. Structured design. *IBM Systems Journal*, 13(2):115–139, 1974.

[17] A. X. Zheng, M. I. Jordan, B. Liblit, and A. Aiken. Statistical debugging of sampled programs. In *NIPS*, volume 16, 2003.

[18] A. X. Zheng, M. I. Jordan, B. Liblit, M. Naik, and A. Aiken. Statistical debugging: simultaneous identification of multiple bugs. In *Proceedings of the 23rd international conference on Machine learning*, pages 1105–1112. ACM, 2006.

Reranking-based Crash Report Deduplication

★Akira Moroo ☆Akiko Aizawa ★Takayuki Hamamoto

★ Tokyo University of Science ☆ National Institute of Informatics

Abstract

Software projects collect and deduplicate vastly numerous crash reports from users to fix bugs efficiently. However, most existing automated methods have performance issues during large-scale clustering. We propose a reranking-based crash report clustering method. Our method is a combination of two earlier methods. By computing similarity used in ReBucket for the crash reports that are highly similar to the query crash report, the method can process reports with throughput equal to that of PartyCrasher. We also introduce an automatically generated dataset for crash report clustering tasks. The evaluation revealed that our method performs at high processing speed while maintaining high accuracy.

1 Introduction

Modern software products such as web browsers have a large and complicated code base along with an increasing number of bugs. To fix these bugs efficiently, developers prioritize high-frequency bugs using crash reports: summaries of the status of the software execution upon its unexpected termination. Figure 1 portrays the structure. A crash report includes a description, machine environment information, and a stack trace. A stack trace presents multiple stack frames with indexes. A lower index denotes a newer frame in the stack trace. An actual crash is caused in the #0 frame: the top frame. Therefore, a bug causing a crash is typically included in the frame near the top frame.

The information in stack traces is helpful for detecting and fixing bugs. Therefore, software-development projects collect such crash report from their users and cluster them according to the detected bugs. However, because of the numerous and diverse crash reports, clustering them manually is infeasible. Many automated crash report deduplication methods have been proposed to address this issue. However, most methods present performance issues because of their high computational costs. As described herein, we propose a reranking-based automated crash report clustering method. Reranking is a well-known technique for information retrieval where the ranking of initial search results is re-ordered using another computationally more expensive ranking function.

The contribution of this paper is that we introduce reranking-based crash report deduplication for reduced computational costs. This report is the first of the relevant literature to describe a study by which a reranking scheme is shown to be effective in crash report deduplication.

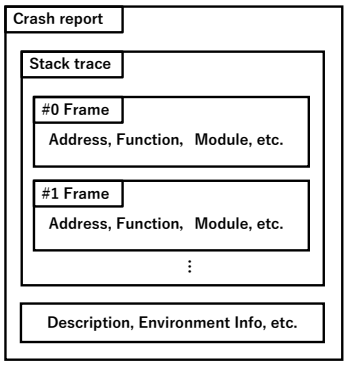


Figure 1: Crash Report Structure.

2 Related Work

Existing methods for automatic crash report deduplication can be categorized into the following two types: One is based on the similarity of feedback from users written in natural language[1, 2]. The other is based on the similarity of stack traces. As described in this paper, we specifically examine the latter type because stack-trace-based methods achieve higher accuracy than NLP-based methods.

Socorro[1] is a crash report management system developed by Mozilla. In this system, a string called *signature* is generated for each crash report by applying several heuristic rules to the stack trace. Then, the system classifies all reports with the same signature into the same group. This system requires human elaboration for editing and updating signature-generation rules.

Lerch and Mezini [3] introduced a method using a TF-IDF-based full-text search engine. Campbell et al. [4] also used a full-text search engine and compared several tokenization methods used for indexing. These methods can achieve reasonable runtime performance for large-scale clustering. However, the accuracy is often worse than that achieved with other methods because the features used in the search engines ignore the order of the frames.

Dhaliwal et al. [5] proposed a method using the Levenshtein distance as similarity between two stack traces. The Levenshtein distance is used to measure the similarity of two strings. This approach regards one frame as a character. Dang et al. [6] defined similarity between two stack

[1]https://wiki.mozilla.org/Socorro

DOI reference number: 10.18293/SEKE2017-135

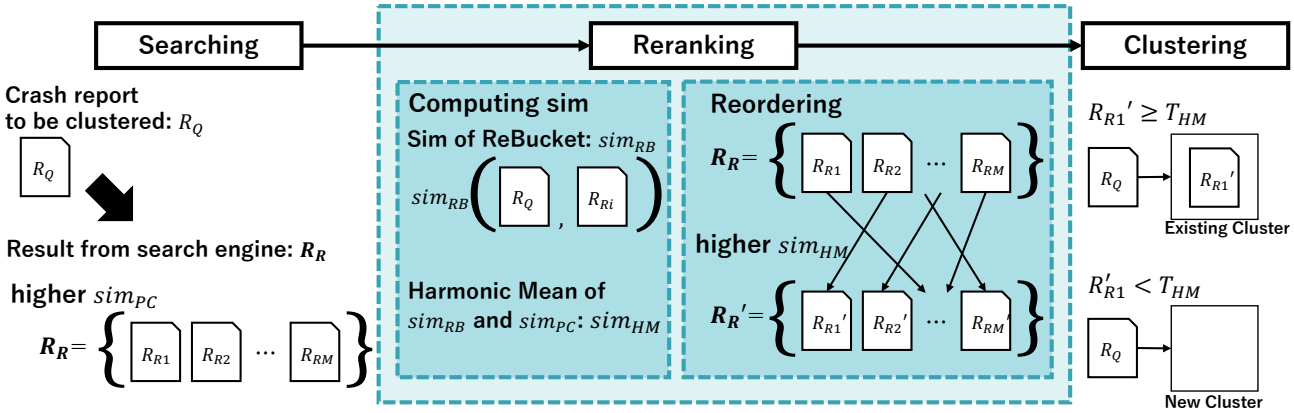


Figure 2: Overview of Proposed Method.

traces based on Position Dependent Model. Because these distance-based methods can capture the features of stack traces, they are useful for high-accuracy clustering but with high computational complexity.

As described in this paper, we use PartyCrasher [4] and ReBucket [6] as baselines. PartyCrasher is implemented as an open-source project[2]. It is easily extensible. Among these methods, ReBucket achieved the highest reported accuracy: 0.88 F1-measure on average.

3 Methodology

The proposed method is a consolidation of two state-of-the-art deduplication methods: ReBucket [6] and PartyCrasher [4]. Figure 2 presents an overview of the method. It comprises three modules: searching, reranking, and clustering. Given a pool of crash reports (already clustered) and a target crash report to be clustered, the search module uses the target report as a search query. Thereby, it obtains an initial ranking list of possibly similar crash reports. We follow the technique proposed in PartyCrasher [4] here, and use a general-purpose search engine. Next, the reranking module re-orders the initial results with the similarity function defined in ReBucket [6]. Finally, the clustering module either selects a cluster from existing ones or generates a new cluster for the target crash report.

3.1 Searching

Let R_Q be a crash report to be processed. Using a full-text search engine, the system obtains the top M search result $\boldsymbol{R}_R = \{R_{R1}, R_{R2}, \ldots, R_{RM}\}$. In [4], multiple tokenization methods for indexing are compared. Our system uses `Camel`, which achieved the highest accuracy. In `Camel`, all frames in a stack trace are tokenized by space separation and for each camel case. Symbols are deleted. Hereinafter, we designate the stack trace of the crash report R_X as C_X.

Most full-text search engines use TF-IDF weight to rank results. Term Frequency (TF) represents how many times a certain term appears in a document. Inverse Document Frequency (IDF) is the reciprocal of the number of occurrences of a certain word in all documents. In our method, we use the same normalization as the one used in Elasticsearch[3]. Letting $W(C_q)$ be a set of words in a stack trace C_q, tf_{t,C_d} and idf_{t,C_d} be a value of DF and IDF of a word t in a stack trace C_d, respectively, and denoting the word count in C_d as nt_{C_d}, the similarity score used in PartyCrasher between C_q and C_d is defined as

$$sim_{PC}(C_q, C_d) = \sum_{t \in W(C_q)} \left(\sqrt{tf_{t,C_d}} \times idf_t \times \frac{1}{\sqrt{nt_{C_d}}} \right). \quad (1)$$

3.2 Reranking

In ReBucket [6], similarity is computed by emphasis on the common frame position between two stack traces. Two metrics are defined: distance to the top frame and alignment offset. The distance to the top frame represents the distance from an arbitrary frame to the top frame in the same stack trace. The alignment offset represents the difference of the distance to the top frame between the frames that match in two stack traces.

These metrics are based on the following insights: First, an important frame has a smaller distance to the top frame. Second, the alignment offset between matching frames is smaller for stack traces with higher similarity. Let f_i and f_j respectively denote the i-th and the j-th frames of C_q and C_d. Denoting distance to the top frame as $min(i, j)$ and alignment offset as $abs(i - j)$, the cost function $cost(i, j)$

[2]https://github.com/naturalness/partycrasher

[3]https://www.elastic.co/products/elasticsearch

of f_i and f_j is defined as follows in ReBucket.

$$cost(i,j) = \begin{cases} e^{-c*min(i,j)}e^{-o*abs(i-j)} & if \;\; f_i = f'_j \\ 0 & otherwise \end{cases} \quad (2)$$

Therein, c and o are weighting parameters. With ReBucket, similarity between stack traces C_q and C_d is defined as

$$sim_{RB}(C_q, C_d) = \frac{M_{m,n}}{\sum_{j=0}^{l} e^{-cj}}, \quad (3)$$

where m and n are at respective depths of C_q and C_d, and $l = max(m,n)$. In addition, $M_{i,j}$ is the optimal cost at depth i,j calculated recursively using dynamic programming.

The computational cost is $O(mn)$ in the above calculation. According to [7], about 90 % of crash reports can be fixed by referring only to the top 10 frames. Therefore, we use only the top 10 frames in our calculation.

The final similarity score between C_q and C_d we use for reranking is given as

$$sim_{HM} = \frac{sim_{RB} \cdot sim_{PC}}{\alpha \cdot sim_{RB} + (1-\alpha) \cdot sim_{PC}}. \quad (4)$$

3.3 Clustering

In the same way as PartyCrasher, we compare the similarity of crash report R'_{R1}, which has the highest similarity in $\boldsymbol{R}'_R$, with the threshold T_{HM}. If it is greater than T_{HM}, then R_Q is clustered into the same cluster as R_{R1}. Otherwise, it is clustered into a new cluster because it is regarded as a crash by the new type bug.

The computational overhead of the search engine in our method is $O(NlogN)$, which is negligible compared with the computational cost of stack trace similarity. Therefore, we compare the size of the distance matrix that requires stack trace similarity calculation. ReBucket uses hierarchical agglomerative clustering (HAC) algorithm for crash report clustering. This method requires $O(N^2)$ because it computes an $N \times N$ distance matrix where N is the number of total crash reports. In contrast, the clustering algorithm used in the proposed method requires $O(MN)$, where M is the number of search results. As with $M \ll N$, the proposed method can reduce the computational cost compared with HAC.

4 Evaluation Experiment

In this section, we present evaluations of PartyCrasher, ReBucket, and our proposed method from the perspectives of accuracy and runtime performance.

4.1 Dataset

We use two datasets: `Launchpad` [4], an existing dataset from Ubuntu Launchpad, and `Firefox48`, a new dataset we collected from Mozilla Firefox. Table 1 presents a summary of the datasets.

Table 1: Dataset Information.

	Project	Crash reports	Clusters	Version	Packages
`Launchpad`	Ubuntu Launchpad	15,293	3,824	N/A	816
`Firefox48`	Mozilla Firefox	36,752	727	from 48.0a1 to 48.0b99	1

`Launchpad` is more reliable than `Firefox48` because `Launchpad` contains clusters produced by hand, whereas the clusters of `Firefox48` are automatically constructed as shown below. `Firefox48` differs from `Launchpad` in that the dataset includes a single software package whereas `Launchpad` includes many different packages provided by Ubuntu's package manager. Many projects deal with a single product in their report systems. Therefore, we presume that our evaluation can be more solid by introducing the `Firefox48` dataset.

We followed [5] to create the dataset, which comprises crash reports and corresponding bugs. A set of crash reports that link to the same bug form a cluster. We gathered 100 crash reports at most from each of the top 300 high-frequency clusters between ver. 48.0a1 and 48.0b99. When the same bug is found to cause two crash clusters, these clusters are grouped into a single cluster. Furthermore, we extract stack traces and metadata from the crash reports.

4.2 Evaluation Metrics

Let BCubed precision be denoted as Bp and BCubed recall as Br. Bp represents how many reports in an estimated cluster belong to the same cluster in ground truth. Br denotes how many reports in a cluster of grand truth belong to the same estimated cluster. The quality of the generated clusters is better when the values of Bp and Br are closer to 1.0. Based on the Bp and Br, the BCubed F1-meature (BCF) is calculated as $BCF = (2 \cdot Bp \cdot Br)/(Bp + Br)$. A tradeoff exists between Bp and Br. Higher BCF indicates higher accuracy[8].

The number of crash reports processed per unit of time is taken as the performance criterion. ReBucket uses multiprocessing to calculate distance matrixes. Therefore, we multiply the actual processing time by the number of processes.

4.3 Determining Parameters

We determined optimal parameters using a subset from the first 20 % of the dataset. The rest are used for testing. The time cost of determining optimal parameters is not included in the performance results. Table 2 presents the final values used in the evaluation. Both ReBucket and our method require c and o in (2). T_{RB} is a distance threshold for HAC in ReBucket. Moreover, α in (3), M, the size of initial ranking list, and T_{HM}, the clustering parameter, are

[4] https://archive.org/details/bugkets-2016-01-30

Table 2: Parameters used in the evaluation.

	c	o	T_{RB}	α	M	T_{HM}
Launchpad	0.3	0.1	0.07	0.3	50	11.0
Firefox48	0.4	0.4	0.26	0.0	50	2.0

Table 3: Evaluation Results of Accuracy.

	Launchpad			Firefox48		
	Bp	Br	BCF	Bp	Br	BCF
PartyCrasher	0.723	0.671	0.696	0.778	0.424	0.549
ReBucket	0.775	0.515	0.618	0.662	0.717	0.688
Proposed	0.807	0.626	0.705	0.728	0.651	0.687

Table 4: Performance Evaluation Results.

	Performance $[Reports/s]$	
	Launchpad	Firefox48
PartyCrasher	2.500	4.663
ReBucket	0.004	0.021
Proposed	2.922	4.022

used only for the proposed method. The number of used processes of ReBucket is 60 because of limitation of the available computation resources. The parameters c, o, and T_{RB} are robust and about 20 % of the searched combinations achieve over 90 % of the highest BCF. By contrast, α, M and T_{HM} are sensitive. Especially, we observe that when M is too large, both accuracy and performance decrease.

4.4 Results

Table 3 presents accuracy evaluation results. The proposed method achieved the highest BCF value for Launchpad. Bp of our method is also the highest. In addition, results show that ReBucket showed high accuracy in terms of BCF because the Br of PartyCrasher drops to 0.424. Moreover, the BCF value of the proposed method is the same as that of ReBucket.

The difference in accuracy between Launchpad and Firefox48 can be attributed to the differences of the numbers of packages: 816 for Launchpad and 1 for Launchpad. It is likely that the same cluster includes reports from the same package in most cases. Results show that ReBucket has a difficulty recognizing the difference when the dataset includes multiple packages because it relies completely on frames in the stack traces.

To validate this point, we created a subset of Launchpad using reports from only one package, "nautilus." Then, the BCF values of PartyCrasher, ReBucket, and our method become 0.616, 0.662, and 0.666, correspondingly. With the subset, ReBucket achieves better performance than PartyCrasher, which is the opposite of the result with the original Launchpad. Based on this result, we conclude that ReBucket does not perform well if multiple packages are included in the dataset. The proposed reranking scheme compensates the issue by filtering out unrelated candidates before applying ReBucket.

Table 4 presents performance evaluation results. Table 4 shows that the processing speed of the proposed method is higher than that of ReBucket. In addition, our method achieved almost equal performance to that of PartyCrasher. Our method is almost 730 times faster.

To summarize, our results demonstrate that the proposed method has better runtime performance than ReBucket while maintaining accuracy.

5 Conclusion

As described in this paper, we propose an automated crash report deduplication method applying reranking against reports of the results obtained from the full-text search engine. We also constructed a new dataset for crash report clustering tasks from Mozilla Firefox. The evaluation results demonstrate that our method is equal to that of ReBucket in terms of accuracy, but with faster performance.

Our current method uses statically tuned parameters with a subset. Manual merging or division of crash clusters is assumed in real operations. In future studies, we expect to investigate a dynamic parameter tuning method using user feedback for more stable accuracy.

References

[1] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," in *Proceedings of the 29th International Conference on Software Engineering*, 2007, pp. 499–510.

[2] A. Alipour, A. Hindle, and E. Stroulia, "A contextual approach towards more accurate duplicate bug report detection," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, 2013, pp. 183–192.

[3] J. Lerch and M. Mezini, "Finding duplicates of your yet unwritten bug report," in *Proceedings of the 2013 17th European Conference on Software Maintenance and Reengineering*, 2013, pp. 69–78.

[4] J. C. Campbell, E. A. Santos, and A. Hindle, "The Unreasonable Effectiveness of Traditional Information Retrieval in Crash Report Deduplication," in *Proceedings of the 13th International Conference on Mining Software Repositories*, 2016, pp. 269–280.

[5] T. Dhaliwal, F. Khomh, and Y. Zou, "Classifying Field Crash Reports for Fixing Bugs: A Case Study of Mozilla Firefox," in *Proceedings of the 2011 27th IEEE International Conference on Software Maintenance*, 2011, pp. 333–342.

[6] Y. Dang, R. Wu, H. Zhang, D. Zhang, and P. Nobel, "ReBucket: A Method for Clustering Duplicate Crash Reports Based on Call Stack Similarity," in *Proceedings of the 34th International Conference on Software Engineering*, 2012, pp. 1084–1093.

[7] A. Schröter, N. Bettenburg, and R. Premraj, "Do stack traces help developers fix bugs?" in *2010 7th IEEE Working Conference on Mining Software Repositories*, 2010, pp. 118–121.

[8] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, "A Comparison of Extrinsic Clustering Evaluation Metrics Based on Formal Constraints," *Inf. Retr.*, vol. 12, no. 4, pp. 461–486, 2009.

Analyzing duplication on code generated by Scaffolding frameworks for Graphical user interfaces

André M. Andrade
Crateús Campus
Fed. Univ. of Ceará
andre@crateus.ufc.br

Rodrigo A. Vilar
Exact Sciences Department
Fed. Univ. of Paraíba
rodrigovilar@dce.ufpb.br

Anderson A. Lima, Hyggo Almeida, Angelo Perkusich
Embedded Systems and Pervasive Comp. Lab.
Fed. Univ. of Campina Grande
anderson.lima,hyggo,perkusic@embedded.ufcg.edu.br

Abstract

Scaffolding is an approach used by some modern web frameworks in order to generate an initial version of applications code based on domain model meta data. Since this temporary code should be customized by programmers to implement real systems, its quality metrics are important aspects. In this paper, a methodology is proposed and applied in order to relate domain model size and a quality metric — amount of duplicated code — focusing on Graphical user interface implementation. Results show that code duplication grows at least linearly with the growth of the number of entities in domain model. There are also some scenarios where quadratic proportions were found. These observations suggest that, for large domain models, code quality and its evolution would be affected when scaffolding frameworks are used.

Keywords - Code generation, Scaffolding frameworks, Code duplication, Meta data

1 Introduction

Automatic generation is an approach that has been widely used on modern web frameworks, e.g. Ruby on Rails, Grails, Yeoman and Spring Roo[1]. These frameworks implement a technique, called Scaffolding [13], with algorithms and templates that generate application initial code and configuration, based on domain model meta data. After generation, developers evolve and maintain source code in order to implement real applications. In doing so, programmers would expect that Scaffolding frameworks produce code with good quality metrics, to ease their work. This means that code smells, like duplication and coupling, should be avoided by scaffolding algorithms.

In this paper, an experiment was designed, performed and analyzed to measure duplication on code produced by two popular web frameworks, Ruby on Rails and Spring Roo, specifically for Graphical User Interfaces (GUI), which represents a great amount of development effort on Enterprise Applications [11, 10]. This study is part of a research project aiming at understanding and enhancing meta data usage on Enterprise Application development. In this case, the proposed experiment design is an innovative approach to analyze how duplication behaves on generated code as software complexity grows.

The results suggest that, in spite of giving advantages for project initial steps like configuration and ramp-up, using scaffolding frameworks can lead to problems on code evolution and maintainability, when applications have complex domain models. Linear and quadratic proportions between domain model size and duplicated code amount were found within this experiment scope, indicating that code quality in applications with vast domain models would degrade.

In Section 2, Scaffolding frameworks and Duplicated Code detection are rationalized. Section 3 details Experiment design for this study, whose results were analyzed in Section 4. Finally, conclusions and future work proposals are presented in the last Section.

2 Background

This section gives an overview of Scaffolding frameworks and duplicated code detectors.

[1] rubyonrails.org, grails.org, yeoman.io, projects.spring.io/spring-roo

DOI: 10.18293/SEKE2017-164

Scaffolding Frameworks

Several modern frameworks generate code automatically based on domain model features, such as, entities, properties, relationships and other meta data [7]. As its name suggests, code generated by Scaffolding frameworks is not designed to be definitive. It is only a first functionality draft that should be polished in order to produce the desired functionality.

Among several frameworks that use Scaffolding, such as, Spring Roo, Play Framework, Apache Tapestry, ASP.NET Dynamic Data, Ruby on Rails, CakePHP, Django and Yeoman, two technologies have been chosen for this work, in order to analyze code duplication on scaffolded applications: Ruby on Rails (RoR) and Spring Roo (Roo). The key factors to choose these frameworks were: relevance due to industrial and worldwide adoption, according to the BuildWith's Framework Usage Statistics [2]; coverage of different language types (static and dynamically typed languages); and availability of code duplication detection tools, in order to operate this experiment.

Ruby on Rails (RoR) allows users to build features quickly using the `generate scaffold` command to generate the application code [6]. Despite being very seductive because of facility and quickness, "the complexity and sheer amount of code in the scaffolding can be utterly overwhelming to a beginning Rails developer" [6]. Generated code consists of Ruby language files (.rb) for back-end and Embedded Ruby files (.erb) for front-end view.

Spring Roo is a domain-driven development framework [8] that uses Java, Spring, AspectJ and Maven[2], combining Java annotations with shell commands to generate all application layers. Roo uses Java on back-end and Java Server Pages XML compliant files (.jspx) on front-end view pages.

Duplicated Code Detectors

Duplication is a strong evidence of high coupled and poor reusable source code. According to Fowler, code duplication represents a conceptual coupling [4], because maintenance on copied code must be done in all code copies. Lee, Barta and Juliff also observed that high coupling and code duplication increase effort for system maintenance [9], including on enterprise applications.

There are several techniques to detect duplicated code in software projects: text-based, token-based, metric-based, AST-based (Abstract Syntax Trees), PDG-based (Program Dependence Graphs) [1]. Two of those techniques were used in this research: AST-based that parses program code into an abstract syntax tree, divides it into sub trees and marks common sub trees as code clones; and text-based that compares every line of code as a string and marks code clones based on the similarity between the text fragments [1].

For Ruby on Rails, three duplicated code detection tools were found: Towelie, Flay and Simian[3]. Flay, which is based on AST, presented better results and is able to process both .rb and erb. files. Therefore Flay was chosen to analyze Ruby on Rails code.

For Spring Roo applications, PMD-CPD[4], which uses String matching, was chosen, because it is one of the most popular static code analysis tool for Java projects. It is able to detect code duplication in Java and JSP files. Since, PMD-CPD does not work with AspectJ files, all aspects code was moved to Java classes, in order to check its duplication. This operation is common for Spring Roo and is performed through a simple command.

3. Experiment Design

In order to analyze duplication on code generated by Scaffolding frameworks, several experiment units have been designed. Essentially, this experiment controls two input variables — domain model characteristics and technology (Ruby on Rails or Spring Roo) — and generates values for one output variable: amount of duplicated code.

Domain Model Scenarios

To control domain model input variable, six scenarios groups were prepared. Table 1 details domain models structure used by each scenario group, specifying quantity of entities and properties. Besides that, scenarios groups explore another domain models features such as property names and types. The most complex scenario is composed by six entities, whose size is similar to small, but useful, systems like: a project management application containing projects, workers, activities, time appointment and artifacts; or a purchase control system containing costumers, vendors, products, sale and sale item.

Scenario groups use a variable that ranges from 2 to 6 (except on scenario F that varies from 1 to 6). In doing so, each scenario group produces several scenario instances as inputs for the experimental units. Therefore, this experiment has 31 scenarios instances that were evaluated using Ruby on Rails and Spring Roo.

In a scenario group, the influence of one domain model feature over code duplication can be tested. In scenario group A, for instance, the number of entities (without properties) varies from 2 to 6. So, code generated by scaffolding frameworks can be analyzed for each scenario instance,

[2] spring.io, eclipse.org/aspectj, maven.apache.org

[3] github.com/gilesbowkett/towelie, github.com/seattlerb/flay, www.harukizaemon.com/simian/

[4] pmd.sourceforge.net/pmd-4.3.0/cpd.html

and code duplication amount can be related to number of entities. In scenario group F, this same procedure can be repeated to relate duplication amount to number of properties in two entities.

G	Description	Variable meaning
A	N Entities without properties	N:Number of entities
B	N Entities with 1 distinct property	N:Number of entities
C	N Entities with 1 property with same name	N:Number of entities
D	N Entities with 1 property with same type	N:Number of entities
E	N Entities with 1 property with same name and type	N:Number of entities
F	2 Entities with M distinct properties	M:Number of properties

Table 1. Experiment scenario groups

Experiment execution

For each experiment unit, which received as input a scenario instance (si) and a scaffolding framework (sf), the following steps were executed: run sf shell commands to generate code concerting si; measure code duplication (with Flay for RoR and CPD for Roo); and register the amount of duplicated code.

It was required some customization in order to tune duplicated code detection. Flay and CPD define minimum threshold values in order to take into account duplicated code blocks. There are default values ($MassThreshold = 18$ on Flay[5] and $DuplicateChunkSize = 100$ on CPD[6]), which can be customized according to user needs.

After running experimental units with $MassThreshold = 18$ for Ruby on Rails and Flay, several relevant duplicated blocks were not detected. Therefore, for this experiment, threshold values were decreased ($MassThreshold = 4$ on Flay and $DuplicateChunkSize = 40$ on CPD), in an effort to maximize code duplication detection.

Measurement units for Flay and CPD vary due to its distinct clone detection approaches. Flay measure similar ASTs by its *mass*. CPD compares repeated string *tokens*. This difference is not a problem because this experiment goal is not to compare RoR and Roo quality. In both cases, the objective is to relate code duplication (either by mass or by token) with domain model size.

[5] docs.codeclimate.com/docs/duplication

[6] maven.apache.org/plugins/maven-pmd-plugin/cpd-mojo.html

4 Result Analysis

In order to organize data results from experiment, several aforesaid factors were used: technology (RoR or Roo); scenarios groups; number of entities (N); and number of properties by entity (M). There is also another determinant to group duplicated code amount: layer. It is because both Ruby on Rails and Spring Roo use the MVC pattern [5] and, since this work focuses on GUI, view and controller layers should be considered. However each layer needs to be evaluated separately because its code uses distinct languages.

G	Variation		Duplicated Code amount			
	N	M	RoR		Roo	
			view	contr	view	contr
A	2	0	1006	308	3010	3311
	3	0	1956	462	3010	5237
	4	0	3206	616	3010	7005
	5	0	4756	770	3010	8773
	6	0	6606	924	3010	10830
B	2	1	1314	312	8273	3311
	3	1	2580	468	10448	5237
	4	1	4254	624	12495	7005
	5	1	6336	780	14796	8773
	6	1	8826	936	16843	10830
C	2	1	1314	312	8273	3311
	3	1	2580	468	10448	5237
	4	1	4254	624	12495	7005
	5	1	6336	780	14796	8773
	6	1	8826	936	16843	10830
D	2	1	1314	312	8273	3311
	3	1	2580	468	10448	5237
	4	1	4254	624	12495	7005
	5	1	6336	780	14796	8773
	6	1	8826	936	16843	10830
E	2	1	1314	312	8273	3311
	3	1	2580	468	10448	5237
	4	1	4254	624	12495	7005
	5	1	6336	780	14796	8773
	6	1	8826	936	16843	10830
F	2	1	1314	312	8273	3311
	2	2	1382	314	8775	3311
	2	3	1450	316	9747	3311
	2	4	1518	318	11516	3311
	2	5	1586	320	13974	3311
	2	6	1654	322	16970	3311

Table 2. Experiment results

On Table 2, where resulting data is shown, each line defines a scenario instance, and columns represent, respectively, scenario group (G), number of entities (N), number of properties (M) and amount of duplicated code for: Ruby

G	Ruby on Rails and Flay				Spring Roo and CPD			
	View		Controller		View		Controller	
	Model	R^2	Model	R^2	Model	R^2	Model	R^2
A	$150N^2 + 200N + 6$	1	$154N$	1	3010	1	$1857.4N - 398.4$	0.999
B	$204N^2 + 246N + 6$	1	$156N$	1	$2148.8N + 3975.8$	0.999	$1857.4N - 398.4$	0.999
C	$204N^2 + 246N + 6$	1	$156N$	1	$2148.8N + 3975.8$	0.999	$1857.4N - 398.4$	0.999
D	$204N^2 + 246N + 6$	1	$156N$	1	$2148.8N + 3975.8$	0.999	$1857.4N - 398.4$	0.999
E	$204N^2 + 246N + 6$	1	$156N$	1	$2148.8N + 3975.8$	0.999	$1857.4N - 398.4$	0.999
F	$68M + 1246$	1	$2M + 310$	1	$328.8M^2 - 563.1M + 8526.4$	0.999	3311	1

Table 3. Polynomials models found for each scenario group, framework and layer

on Rails (RoR) view and controller; and Spring Roo (Roo) view and controller.

This experiment executed 62 experimental units, which were designed after combining all possible input variables: 31 domain model scenarios (explained in Section 3); and 2 technologies (Ruby on Rails and Spring Roo).

Data synthesis

Some function models were studied in an attempt to express, based on experiment results, a relation between amount of duplicated code and domain model complexity, that is number of entities (N) and properties (M). Therefore, polynomial regression model was used to find the best mathematical function composed of an n-degree polynomial that fits the relation between CD and M or N, such as:

$$CD \approx f(N) \text{ or } CD \approx f(M)$$

where function f is a polynomial with degree n.

Polynomial model reliability is analyzed from residuals that define a coefficient of determination (R^2) [3], which reflects the correlation between the resulting samples from CD and the values of M and N. The closer this number is to 1, the better is the polynomial model.

CD function models have the following structure:

$$CD_{g,s,l}(N) = f(N) \text{ or } CD_{g,s,l}(M) = f(M)$$

where g is scenario group (A, B, C, D, E or F); s is scaffolding framework (RoR or Roo); l is layer (Controller or View); N is number of entities; M is number of properties by entity; $f(N)$ and $f(M)$ are n-degree polynomials. The code duplication detection tool – Flay or CPD – was not considered as a parameter since they do not vary in the same framework.

For each executed combination of framework, scenario group and layer, the polynomial that best fits to CD samples was chosen, according to R^2 value, and having lower degree. Table 3 shows the polynomials chosen for each scenario group. Considering same combination of framework and layer, the B, C, D and E scenarios showed the same polynomial model, therefore, their graphs are represented in the same line on the Table 4, which represents graphically the experiment results.

Interpretation

Based on the polynomial models found and their related graphs, some observations can be defined, within this experiment scope:

1. Code duplication in Ruby on Rails view layer varies in quadratic proportion to number of entities and in linear proportion to number of properties, with perfect models found;
2. Code duplication in Ruby on Rails controller layer varies in linear proportion to both number of entities and number of properties, with perfect models found;
3. Code duplication in Spring Roo view layer is constant for any number of entities without properties (scenario group A), with a perfect model found;
4. Code duplication in Spring Roo view layer vary in linear proportion to number of entities (with properties) and in quadratic proportion to number of properties, without perfect models found;
5. Code duplication in Spring Roo controller layer is in linear proportion to number of entities, without perfect models found;
6. Code duplication in Spring Roo controller layer is constant for any number of properties in two entities (scenario group F), with perfect a model found;
7. For Ruby on Rails all models found have $R^2 = 1$ i.e., polynomials have high quality to represent experiment scenarios and to possible predict code duplication for greater domain models;
8. For Spring Roo all non constant models have $R^2 = 0.999$. In fact, several polynomials would represent this relation with $R^2 = 0.999$ and the one with lowest

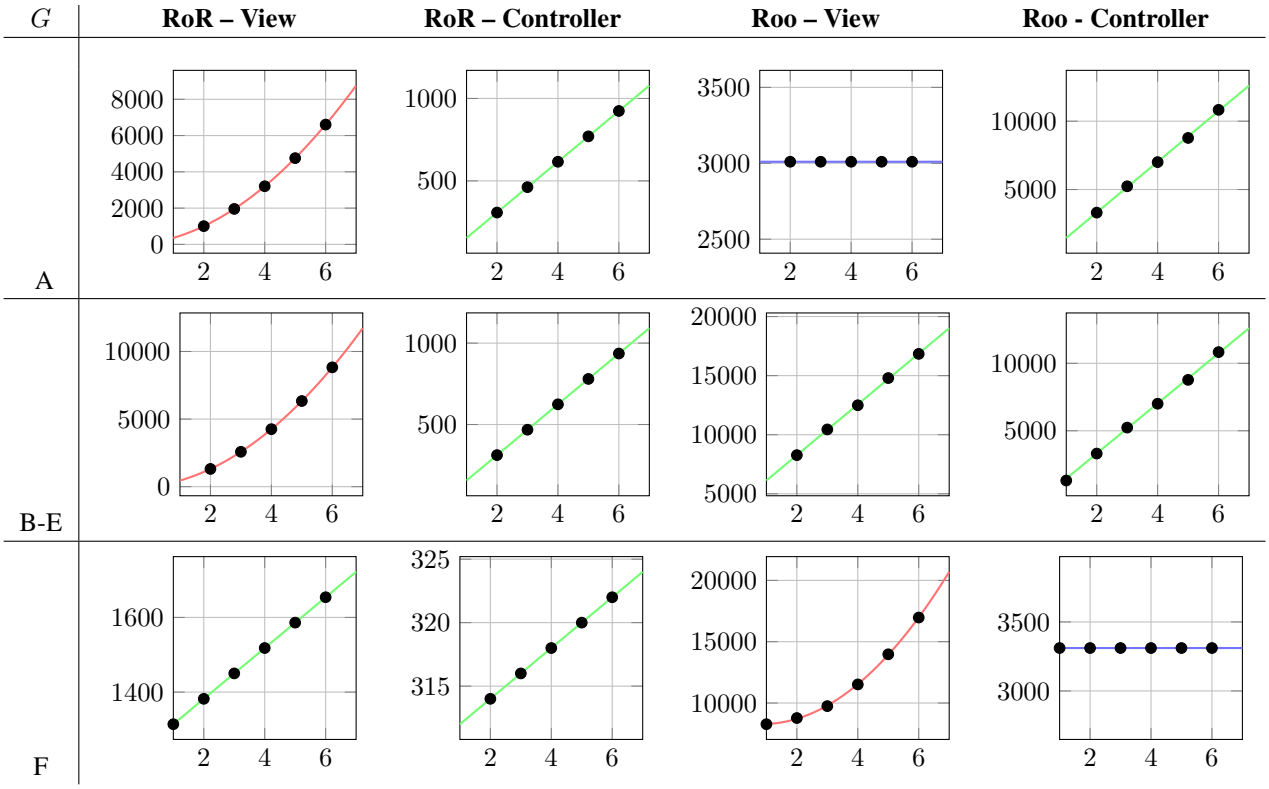


Table 4. Polynomials plots for all scenario groups

degree was chosen. In spite on being very good models, they can contains errors and should be validated in other experiments with greater number of samples.

Ideal frameworks should generate code without duplication or, at least, with a constant amount of duplicated code. In this experiment, two observations define constant relations, however these observations are improbable to occur on real systems. Observation 3 represents entities without properties, which is a strange scenario for real projects, because every entity should have properties. Observation 6 takes place on systems with only two entities. In spite of being very small and improbable systems, this result is useful because it indicates that number of properties does not influence code duplication in Spring Roo controllers.

After combining all observations, this experiment shows that code duplication grows in linear or quadratic proportion to number of entities or properties, for all scenarios, considering view and controller layers together.

These relations predicts that, for two completely different scaffolding frameworks, generated code is duplicated in a significant amount. At least, code duplication grows in a linear proportion to domain model size.

Models with $R^2 = 0.999$ have several polynomial models to represent code duplications on units of this experiment. However, in this work, it is not possible to define the most appropriate function. Therefore, replications of this experiment could be made to obtain clearer conclusions for scenarios with N and M greater than 6.

Threats to validity

It's necessary to consider the risk of the many generalizations found here: from small domain models to large domains; from two scaffolding frameworks to all scaffolding frameworks; and from two duplicated code detection tools to all detection tools.

Due to time and resource restrictions, only small domain models were analyzed. However a great effort was made in this work to create an analysis methodology that can be easily reproduced. In doing so, the observations found here can be tested for larger domains and with other technologies.

In spite of using few scaffolding frameworks, the chosen frameworks use different approaches, language types and clone detection techniques. These differences enhance results strength.

There is also a lack of scenarios that increase both entities and properties quantity. In doing so, the impact of number of entities and properties combined was not tested in this work.

Packaging

All experiment code and scripts are available on two open source repositories, one for Ruby on Rails[7] and the other for Spring Roo[8]. Each scenario group is organized into separated branches and there is a commit for each experiment unit.

These repositories contains a README file on master branch that explain how to reproduce and expand this experiment.

5 Conclusion

This paper presents an approach that is, as far as we know, innovative to relate domain model complexity and duplication on code generated by scaffolding frameworks.

For domain models with up to six entities and properties, duplicated code results could be generalized into polynomials, which lead to two main conclusions: (i) code duplication increases, at least, at linear proportion to domain model size, on GUI generated by two scaffolding frameworks (Ruby on Rails and Spring Roo); and (ii) for models that fitted on experiment results with high quality, it could be expected that the same models would work for greater domain models.

The consequence of conclusion (i) is that duplicated code on applications generated by scaffolding framework can increase to a severe amount when it has complex domain models, and this increase would inhibit code evolution.

Conclusion (ii) can be tested in bigger scenarios and, if it is confirmed, the polynomials found by this work would be used to estimate code duplication, on code generated by Ruby on Rails and Spring Roo, for applications with complex domain models.This estimation would also help software architects to choose the scaffolding framework that will be used in development projects.

As future works, we suggest to replicate this experiment with other Scaffolding frameworks, such as, Grails (Groovy), django (Python), CakePHP, ASP.NET; and with bigger domain models.

Besides that, for frameworks that easily expose its templates, such as, Ruby on Rails, asymptotic computational complexity analysis on code generation algorithms could be compared to our experiment results, in order to confirm or deny them.

In order to analyze the most popular GUI frameworks approaches, identify its pros and cons related to software maintenance, and propose better solutions, this research is the second part of a bigger work that had first results published at 2015 by our coauthor Rodrigo Vilar. Vilar [12] already showed good results by using some rendering patterns to provide reuse on GUI code. As next step, we intend to present a framework proposal and a comparative analysis with popular GUI frameworks considering the relation between complexity, productivity and maintainability.

[7] github.com/Andersoonalves/Rails_Duplicate_Code_Analysis

[8] github.com/rodrigovilar/RooCodeDuplication

References

[1] Y. L. Aritra Ghosh. An empirical study of a hybrid code clone detection approach on java byte code. *GSTF Journal on Computing (JoC); Singapore*, 5(2):34–45, 2017.

[2] BuildWith. Framework usage statistics: Statistics for websites using framework technologies. *https://trends.builtwith.com/framework*, 2017.

[3] G. C. R. Douglas C. Montgomery. *Applied Statistics & Probability for Engineers*. Wiley, 3 edition, 2002.

[4] M. Fowler. Reducing coupling. *IEEE Software*, (4):102–104, 2001.

[5] M. Fowler. *Patterns of enterprise application architecture*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[6] M. Hartl. *Ruby on Rails Tutorial: Learn Web Development with Rails*. Addison-Wesley Professional, 2016.

[7] J. Herrington. *Code generation in action*. Manning Publications Co., 2003.

[8] S. M. Josh Long. *Getting Started with Roo. Rapid Application Development for Java and Spring*. O'Reilly, 2011.

[9] M. Lee, B.-Z. Barta, and P. Juliff. *Software quality and productivity: theory, practice, education and training*. Springer, 2013.

[10] G. Meixner, G. Calvary, and J. Coutaz. Introduction to model-based user interfaces - w3c working group note 07 january 2014.

[11] B. A. Myers and M. B. Rosson. Survey on user interface programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 195–202. ACM, 1992.

[12] D. O. Rodrigo Vilar and H. Almeida. Rendering patterns for enterprise applications. In *Proceedings of the 20th European Conference on Pattern Languages of Programs*, EuroPLoP '15, New York, NY, USA, 2015. ACM.

[13] D. Thomas and D. Hansson. *Agile Web Development with Rails*. Pragmatic Bookshelf, 2006.

Dedicated Static Sampling Pointcut Designators

Amjad Nusayr
University of Houston Victoria
Victoria, TX, 77904, USA
Email: nusayra@uhv.edu

Abstract— Sampling-based instrumentation is often used to make runtime monitoring of applications more efficient. AOP has long been used for monitoring but does not have the ability to support code sampling in its current form. In this paper, we present an implementation of two new pointcut designators in a static weave form intended to be used as code sampling tools. The implementation is created as an extension using the aspect bench compiler (abc). These two new pointcut designators serve as additions to the family of joinpoints in AOP for the purpose of code sampling.

I. INTRODUCTION

Aspect Oriented Programming (AOP) is mainly concerned with the separation of cross-cutting concerns; the design of AOP was dedicated to separate the program main logic functionally from secondary yet; imperative code additions that otherwise make the code weak or susceptible to break [9]. Take for example the code needed for a student to register in a class. The basic logic of this code is simple; first make sure that the class still has seating space for the student. If so, add the student to class roster. Now creating the code for the above logic is straight-forward, but does not represent reality. In real life, there should be a segment in the code that always checks to authenticate the user during the session. There should be as well code that logs any transactions to some database. These two actions are not part of the basic program logic but should be incorporated to validate the program and the actions of the user.

Another major example is code based exceptions. The mere semantics of an exception is to catch some behavior that is not covered and does not meet the basic program logic. The simple example of "DivideByZero" exception shows the importance of having such cross-cutting concerns implemented in code.

The concept of cross-cutting concerns in AOP has been also utilized in other arenas. The research shows much usage of AOP in domain specific languages [6, 7]. This usage spanned from expressing constraints on the behavior and structure of a program all the way to parallel dynamic analysis of multicores [2]. The growth of AOP has itself been extended to include more sophisticated tools in aiding code development. This paper describes two new sampling designators, their main functionality, their implementation, and how they are used. It also presents a performance analysis compared to other designators.

II. RELATED AND BACKGROUND WORK

Since AOP was established, the notion to extend the set of pointcut designators was defined [3, 13]. AOP introduces a modular approach in helping to separate crosscutting concerns by an *Aspect*. An aspect relies on the concept of *weaving*; the instrumentation of crosscutting code into the main program code logic. A typical aspect module contains several fundamental components; the first component is the *pointcut designator*. Pointcut designators are features in the program execution where the advice of an aspect can be weaved in. A composition language allows a pointcut expression to combine and constrain these to define a pointcut (execution occurrences of the program features) that satisfies the expression and where the advice will be executed[1]. The *advice* represents the code to be weaved at locations defined by the pointcut expression.

The code below shows an example of an *after* advice that is weaved after every time the method openSocket is executed. The advice will print out the reflective information about, where in the code the particular execution has occurred. Note that in this particular example, the advice will apply on any method that begins with "openSocket" regardless of the return type and the number of parameters.

```
after() execution(* openSocket*(..)) {
    //Advice to be woven
    System.out.println("socket open at"+
    thisJoinPoint.getSourceLocation());
}
```

The amount of AOP integration within a system depends on the way it is implemented. It could simplify the base code and the semantics of the program or it could do the opposite by introducing too much coupling among different aspects. The latter defeats the purpose of AOP. A well-known study examines the interactions between aspects and methods and identifies classes of interactions that enable modular reasoning about the crosscut program [12].

AOP has been implemented to be used in several languages. One of the most used is AspectJ; an implementation of AOP for the Java language [8]. An alternative to AspectJ is the aspect bench compiler (abc) [3]. The main difference between AspectJ and abc is that abs is designed for extensibility and modification by creating extensions. The extension of abc includes introducing new grammar to the language, followed by introducing new rules for the semantics and code needed for abstract syntax trees. Adding advice includes defining how code

1. AOP literature uses the term joinpoint to refer to a point in the code execution. We feel that the term "pointcut designator" has a more abstract meaning and usage.

DOI reference number: 10.18293/SEKE2017-175

should be generated to invoke that advice and where in the joinpoint shadow this code should go. The Shadow pointcuts pick out a specific joinpoint shadow within a method body (i.e. enables to shadow every executable statement in the code).

Sampling could be defined as a general utility of summarization for a broad spectrum of analytical tasks [4]. In the world of computer science, sampling research is usually conceived to target data sampling techniques. An example is sampling big data and graph, and network sampling. BlinkDB,is a well-known approximate query engine on large quantities of data used for sampling [1, 4].

The use of the term *code sampling* is generally found in compiler research [5]. Code sampling is used to determine many aspects of the code such as unreachable code fragments (with the help of code coverage), dead-code elimination, and the replacement of frequent accessed code segments with other faster segments.

III. WHAT TO SAMPLE

The first set of questions the think of are, A) which locations or points in a program should be selected for sampling? B) what exactly should be sampled? And C) how often should it be done? These three questions were at the basic design of the new pointcut designators. The following is a discussion for each question:

A. *Which locations or points in a program should be selected?*

AOP defines a joinpoint as a point in the execution of the program. The abstract norm of the word "point" refers to a location in code. That location could be a decision structure or a loop back-edge or any executable statement. In previous work, we defined the point to be three possibilities [10]: *Code*: In this case, the weaving is based on a particular point or set of points that exist in code. *Time*: Using time as a point means that sampling is controlled by some kind of timing theme. This could be relative time or wall-clock time. *Data*: Using data as the definition of a point means that sampling would be over some data space and weaving runs when this data are accessed and the sampling criteria is met. In this paper, the focus is on code space sampling since sampling is done in a static manner (at compile time).

B. *What exactly should be sampled?*

The sampling process in general can occur in one of two ways; the first is *static sampling*. This means that the selection of the actual points (locations) in code for sampling is going to happen at compile level. This could so abstract and general; for example, sampling for every nth executable statement in code. It could be more detailed by picking every 30% of switch statements in the code. The second type of sampling is *dynamic sampling*, which will weave code that matches some point of execution at runtime (while the program is running). That is the weaving itself happens at runtime. There is no predetermined location in the code to be examined. Dynamic sampling will incur some overhead. This overhead is due to the selection point and the weaving process. The focus in this paper is only on static weaving as dynamic weaving was discussed in other work.

C. *How often should it be done?*

Since static weaving is the choice, one has to be careful on the frequency on selecting the weaving points in code. The decision whether or not to instrument a particular point in the program is made at program compile time.

IV. STATIC WEAVING

In static sampling the decision whether or not to instrument a particular point in the program is made at program generation time. In AOP this translates to weaving-time decisions about whether a joinpoint shadow should match the pointcut expression or not. Our sampling pointcuts essentially take the set of joinpoint shadows that would match the non-sampled pointcut expression and reduce it by the selected mechanism. In this section we introduce two new static weaving pointcut designators; weaveProbability(p) and sliceOf(N,M).

A. *The weaveProbability Pointcut Designator*

The first new static pointcut designator is

weaveProbability(P)

This designator controls whether weaving at a statically matching joinpoint shadow occurs. With probability P the advice will be woven, while with probability 1-P it will be skipped and not woven. This will then correspond to a sampling of the joinpoints at runtime, but where all joinpoints of a given shadow will be executed or not, depending on the compile-time probabilistic decision. Since this is a compile-time decision, the probability used does not automatically correspond to an equal expected runtime reduction. Only if the execution of the joinpoint shadows is relatively uniform will the static probability actually result in a similar dynamic probability. If a small number of joinpoint shadows incur most of the advice execution for a particular pointcut expression, then the runtime overhead will vary greatly, depending on how many of those high-execution shadows have been probabilistically selected.

This pointcut designator is simple and efficient, and is useful for reducing monitoring cost where there is a reasonably large number of static joinpoint shadows and the expected behavior over them is mostly uniform. If there are only a few shadows (e.g., say a program has three sites that match call(Table.insert)), then this sampling pointcut designator would probably not be useful

B. *The sliceOf Pointcut Designator*

The more involved, and perhaps more useful, static pointcut designator is

sliceOf(N,M)

This designator, rather than being probabilistic and thus uncontrollable, is deterministic in its sampling behavior. This designator selects the M_{th} fraction of joinpoint shadows, of size $\frac{1}{N}|JPSet|$ In other words, of the total number of shadows that match the pointcut expression, this selects 1/N of them, and the slice that it selects is the M_{th} one. For example, if 100 joinpoint shadows match a pointcut, then sliceOf(10,1) will select the first 10, sliceOf(10,2) will select the second 10, and sliceOf(10,10) will select the last 10. Fig. 1 shows the sampling of the first 10%

of shadow matches that could be used to study the behavior of program startup

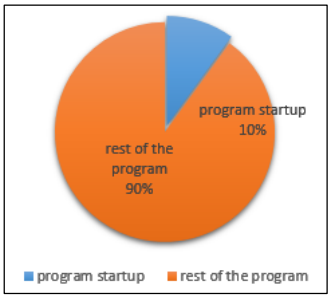


Figure 1: sliceOf(10,1) of a 100 shadow matches applied with the basicblock designator

This pointcut designator allows control over how the distribution of sampling will occur. It enables the creation of N versions of the application, over which it is guaranteed that all shadows that match the pointcut expression are covered, and thus all joinpoints that occur in execution are also covered. These versions can then be deployed and used, and the data collected be used in aggregate to understand some aspect of the program in its entirety.

This designator requires knowledge of how many joinpoints match the pointcut expression in order to slice properly. In the abc framework this is not known until the end of compilation, and so this pointcut designator requires a 2-phase implementation. When the program and aspect is first compiled, it will generate a data file that contains the count of joinpoint shadows. On a re-compile it then uses the data file to perform the slicing and weave the correct shadow sample.

This type of instrumentation is typical of schemes devised for end-user remote application monitoring, where there are competing desires to monitor the whole program but also to not affect the performance for any one user [11]. By distributing different versions of the instrumented program to different users, and collecting and aggregating the obtained data, information about the whole program can be obtained. This scheme also serves to provide some level of privacy assurance to users, since for any particular user only a small portion of application behavior might be recorded.

V. Baseline Measurements and Evaluation

The program Image2Html was used as a small example to first demonstrate sampling and to obtain baseline performance measurements. This program converts a JPEG image into textual HTML-enhanced "ASCII art".

This test used the *basicblock* designator [10] combined with one of the sampling designators; e.g., the pointcut expression is like "before: basicblock() && weaveProbability(.3)". The advice is a simple basic block profiler that counts how many times each block has executed

Fig. 2 shows these results from Image2Html The horizontal lines are baselines of performance of the program with full instrumentation (no sampling) and with no instrumentation. In the fully instrumented version, advice gets executed 3,588,277 times over 411 basic blocks in the program. The three linearly increasing lines on the graph are three dynamic sampling designators which are not in the scope of this paper.

The static sampling shows good and expected performance. The Figure shows datapoints for weaveProbability of 0.15, 0.25, and 0.6; and shows data points for sliceOf for N = 2, N = 4, and N = 8. Since runtime costs for static weaving can change when the weaving is re-applied, each data point shows an average time plus error bars indicating the minimum and maximum observed run times. For weaveProbability, the configuration was compiled and ran four times, and for sliceOf the configuration was compiled and ran for each possible M value (e.g., for N = 8, with 8 tests, with M = 1, 2, 3, ...8). The N = 8 data point is at the 12.5% sampling position on the X axis, while the N = 2 data point is at the 50% sampling position.

Because sampling decisions are made statically, with no runtime check, their performances are in general better than dynamic sampling. As predicted, however, static sampling can suffer from high variation in runtime costs. In this program, weaveProbability does not incur too high of a variation, but sliceOf does. One configuration in N = 8 incurs almost half of the full instrumentation cost, while other incur almost no cost. The range of runtime costs for N = 4 span almost the full range between the no instrumentation and full instrumentation baselines. Finally, the two runs for N = 2 (one being the minimum and one being the maximum) are also quite far apart in runtime cost.

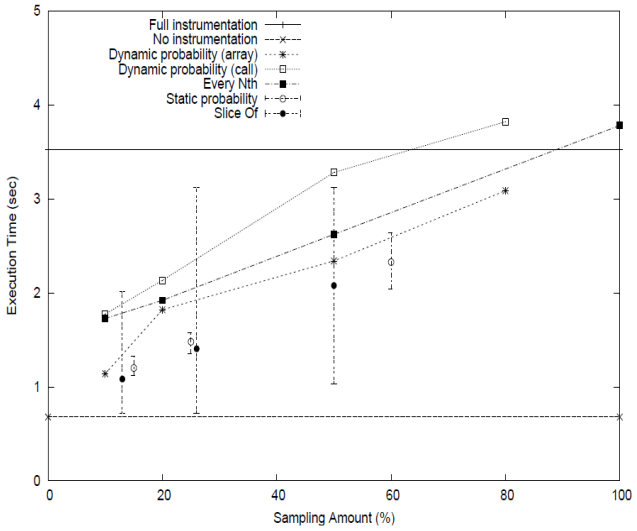


Figure 2: Basic block sampling performance for Image2Html. Error bars on the static pointcut designators (static probability and slice of) show the minimum and maximum of the averaged values

VI. Implementation Issues

The AspectJ documentation separates the fundamental pointcut designators into three categories: kinded, scoping and context [8]. Their documented definitions are: Kinded designators are those which select a particular kind of join point; for example: execution, get, set, call, handler. Scoping designators are those which select a group of join points of interest (of probably many kinds); for example: within, withincode. Contextual designators are those that match (and

optionally bind) based on context; for example: this, target, @annotation.

Each of the three categories above can select a set of joinpoints to execute the advice on, although in practice pointcut expressions usually have at least one kinded pointcut designator. Our sampling pointcut designators are different in that they are meant to operate on an already selected set of joinpoints. The static sampling designators are expected to effect the size of this set, but only in a negative manner: they reduce the given set by some amount.

Typically, new pointcut designators created using abc would perform their own "shadow matching" where they selected some execution points (joinpoint shadows) to include in the joinpoint set (thus including all joinpoints which occur at those shadows). The internal pointcut expression evaluator would then combine with the shadows from other parts of the expression to compute the actual set of joinpoint shadows for the given pointcut expression.

For our sampling extensions, we did not add any shadow matching in the extension, but rather we hook into the optimization phase of pointcut expression evaluation, and it is in this phase that we apply the desired sampling effect. This is achieved by the removal of some of the joinpoint shadows from the selected set.

VII. FUTURE WORK

The basic sampling mechanisms we presented in this paper can be extended in a variety of directions. For the static sampling pointcut designator sliceOf, a better mechanism to control expected runtime costs is needed. One way to do this is with profiling information delivered to the compilation phase. A fully instrumented version could be generated, and under test conditions an expected profile of per-joinpoint-shadow execution costs could be obtained. This would then be used to allocate the expected higher cost shadows among different slices, which should help alleviate the high variation in the performance of the different slices.

Another useful extension would be that of linked sampling, where for example, if probabilistic sampling chose to execute the advice on a top-level method, then all advice (instrumentation) on code that is invoked from that method should be executed as well. For example, if the top-level method invocation represents the handling of one transaction in a TPS, then we could implement sampling on transaction, where all instrumentation would execute for a selected transaction, and not for others.

VIII. CONCLUSION

We presented new pointcut designators for AspectJ with the use of the aspect bench compiler that allow the creation of sampling-based instrumentation, a common need in program monitoring scenarios. With sampling, AOP can be used for efficient monitoring instrumentation even on heavily used code that would otherwise be prohibitive to monitor. Our research goal for this line of work is to enable the high-level formalisms of AOP to be useable for the wide variety of low-level sampling needs. As the future work discussion shows, there is still work left to be done in this venue, but the sampling pointcut designators described in this paper are a significant contribution in this direction.

It is likely that the sampling designators could be useful for other application and system needs. One example would be to sample transaction information of an ecommerce to obtain a real-time statistical profile of user requests. It is likely that developers could think up wide and novel uses for AOP with sampling.

REFERENCES

[1] Albattah, W. 2016. The Role of Sampling in Big Data Analysis. Proceedings of the International Conference on Big Data and Advanced Wireless Technologies (New York, NY, USA, 2016), 28:1--28:5.

[2] Ansaloni, D., Binder, W., Villazón, A. and Moret, P. 2010. Parallel Dynamic Analysis on Multicores with Aspect-Oriented Programming. 2010 Conference on Aspect-Oriented Software Development (AOSD) (2010), 1–12.

[3] Avgustinov, P., Christensen, A.S., Hendren, L., Kuzins, S., Lhoták, J., Lhoták, O., de Moor, O., Sereni, D., Sittampalam, G. and Tibble, J. 2005. Abc: An Extensible AspectJ Compiler. Proceedings of the 4th International Conference on Aspect-oriented Software Development (New York, NY, USA, 2005), 87–98.

[4] Cormode, G. and Duffield, N. 2014. Sampling for Big Data: A Tutorial. Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (New York, NY, USA, 2014), 1975.

[5] Debray, S.K., Evans, W., Muth, R. and De Sutter, B. 2000. Compiler Techniques for Code Compaction. ACM Trans. Program. Lang. Syst. 22, 2 (Mar. 2000), 378–415.

[6] Hadas, A. and Lorenz, D.H. 2015. First-class Domain Specific Aspect Languages. Companion Proceedings of the 14th International Conference on Modularity (New York, NY, USA, 2015), 29–30.

[7] Hadas, A. and Lorenz, D.H. 2016. Toward Disposable Domain-specific Aspect Languages. Companion Proceedings of the 15th International Conference on Modularity (New York, NY, USA, 2016), 83–85.

[8] Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J. and Griswold, W.G. 2001. An Overview of AspectJ. Proceedings of the 15th European Conference on Object-Oriented Programming (London, UK, UK, 2001), 327–353.

[9] Mens, K., Lopes, C.V., Tekinerdogan, B. and Kiczales, G. 1998. Aspect-Oriented Programming Workshop Report. Proceedings of the Workshops on Object-Oriented Technology (London, UK, 1998), 483–496.

[10] Nusayr, A. and Cook, J. 2009. Using AOP for detailed runtime monitoring instrumentation. Proceedings of the 7th International Workshop on Dynamic Analysis -- WODA'09. (2009), 8–14.

[11] Orso, A., Liang, D., Harrold, M.J. and Lipton, R. 2002. Gamma System: Continuous Evolution of Software After Deployment. Proceedings of the 2002 ACM SIGSOFT International Symposium on Software Testing and Analysis (New York, NY, USA, 2002), 65–69.

[12] Rinard, M., Salcianu, A. and Bugrara, S. 2004. A Classification System and Analysis for Aspect-oriented Programs. SIGSOFT Softw. Eng. Notes. 29, 6 (Oct. 2004), 147–158.

[13] Ubayashi, N. 2004. An AOP Implementation Framework for Extending Join Point Models. Proceedings of ECOOP 2004 Workshop on Reflection, AOP and Meta-Data for Software Evolution (RAM-SE'04 (2004), 71–81.

A Publish-Subscribe based Architecture for Testing Multiagent Systems

Nathalia Moraes do Nascimento[⋆] Carlos Juliano Moura Viana[†] Arndt von Staa[⋆]
Carlos José Pereira de Lucena[⋆]

[⋆]Software Engineering Lab (LES),
Pontifical Catholic University of Rio de Janeiro,
Rio de Janeiro, Brazil
nnascimento, arndt, lucena@inf.puc-rio.br

[†]Tecgraf Institute,
PUC-Rio,
Rio de Janeiro, Brazil
cviana@tecgraf.puc-rio.br

Abstract - *Multiagent systems (MASs) have been applied to several application domains, such as e-commerce, unmanned vehicles, and many others. In addition, a set of different techniques has been integrated into multiagent applications. However, few of these applications have been commercially deployed and few of these techniques have been fully exploited by industrial applications. One reason is the lack of procedures guaranteeing that multiagent systems would behave as desired. Most of the existing test approaches only test agents as single individuals and do not provide ways of inspecting the behavior of an agent as part of a group, and the behavior of the whole group of agents. Accordingly, we modeled and developed a publish-subscribe-based architecture to facilitate the implementation of systems to test MASs at the agent and group levels. To illustrate and evaluate the use of the proposed architecture, we developed an MAS-based application and performed functional and performance ad-hoc tests.*

Keywords – group test; agent test; multiagent system; test architecture; publish-subscribe; RabbitMQ

1. Introduction

Multiagent systems have been applied to a wide range of application types, including e-commerce, human-computer interfaces, network control, air traffic control and diagnosis [1] [2]. However, few of them have been commercially deployed [2]. According to Pěchouček and Mařík [2], one reason is the lack of procedures guaranteeing that the distributed systems would behave as desired. In addition, agent-based systems involve different characteristics, such as autonomy, asynchronous and social features, which makes these systems more difficult to understand. Thus, more elaborate methods of verification and testing [3] of multiagent operations should be provided [2].

According to Nguyen et al. (2009) [4], a full testing process of a multiagent system consists of five levels: unit, agent, integration (or group), system and acceptance. Agent test tests the capability of a specific agent to fulfill its goal and to sense and affect the environment. Integration test tests the interaction of agents and the interaction of agents with the environment, ensuring that a group of agents and environmental resources work correctly together [4].

As discussed by Serrano et al. (2012) [5], several approaches have been proposed to test multiagent systems at the unit and single agent levels [6] [7] [8] [9] [10], while there are few studies that address the issue of testing a MAS at group level [6] [11] [5] [12]. In addition, to perform group tests, most approaches have focused on capturing and visualizing messages exchanged among agents. They do not provide ways of also tracking the behaviors of two or more agents in the same view and finding a correlation between their behaviors. For example, Serrano et al. (2012) [5], which is one of the most recent papers published about testing MASs at the group level, uses ACLAnalyser [13], a tool for debugging MAS through the analysis of ACL [14] messages. Thus, by using these current test approaches, if an agent exhibits unexpected behavior (failure), a developer has to inspect this failed agent or messages exchanged between agents to find the fault that caused that failure. However, if an agent fails, its failure may be related to a previous and an unexpected behavior of another agent in the environment. It would be a real problem to some MAS-based approaches, such as that one proposed by Malkomes et al. (2017) [15], which promotes the development of cooperative agents without using message communication.

In the general context of distributed systems, Araújo and Staa (2014) [16] also faced the problem of testing a group of asynchronous components. They realized that most approaches to detect error and diagnose a failure in distributed systems rely on distributed log files over various machines, which makes the comprehension of the interaction among the machines more difficult. Thus, Araújo and Staa (2014) [16] proposed the use of a central architecture to receive, store and inspect timestamp-based logs from distributed machines, enabling a developer to further diagnose failures in a single machine and in the whole system during the software development cycle. Further, they presented a diagnosing mechanism based on logs of events annotated with contextual information, allowing a specialized visualization

DOI reference number: 10.18293/SEKE2017-050

tool to filter them according to the maintainer's needs. However, the authors discuss some limitations in their approach, such as the query response time that grows as the database size grows, which impacts the inspection interface usage, and the use of an inefficient solution for creating discarding rules.

In this paper, we present an architecture that was implemented[1] to make feasible the implementation of agent and group test activities within the MAS software development process. Our approach is based on the architecture to test distributed systems proposed by Araújo and Staa [16]. Nonetheless, MASs involve some characteristics that are not addressed by non-agent-based systems, such as autonomy and social behaviors. Thus, our goal is to adapt the architecture proposed by Araújo and Staa [16] to create one for testing MASs at different levels. Therefore, in order to represent MAS properties, we changed some tags that are used in their approach to represent logs with meta-information annotations. In addition, to solve the problems of data volume and discarding rules presented in their architecture, we decided to use a publish-subscribe [17] technology, instead of a database one. Through a publish-subscribe based approach, it is possible to develop decoupled and different tests that select logs that are useful for their purposes and ignore the irrelevant ones.

To illustrate and evaluate the use of the proposed architecture for creating systems to test MASs, we used and tested a simple MAS-based application. This experiment is presented in section 2. The remainder of this paper is organized as follows. Section 3 introduces the test architecture. Section 4 evaluates the test architecture, presenting the experimental results and evaluation. The paper ends with conclusive remarks in Section 5.

2. APPLICATION SCENARIO

In order to evaluate our proposed approach to test multiagent systems, we developed a simple multiagent application. This application is based on a scenario commonly used in the MAS literature[14] - a marketplace to buy and sell books on-line. We believe this experiment will assist one to understand our approach and facilitate further comparisons and analysis. We developed this application by using the JAVA Agent Development Framework (JADE) that is a Java software framework implemented to facilitate the development of multiagent systems [14].

2.1. Sellers and Clients

This application implements a simple marketplace where users create autonomous agents to sell and buy books for them, as described in the JADE Guide [14]. Therefore, this scenario contains two kinds of agents: Seller and Client. As part of the JADE platform, there is also a Directory Facilitator Agent (DF) that provides a Yellow Pages service by means of which an agent can find other agents providing the services he requires [14]. This illustrative scenario is depicted in Figure 1.

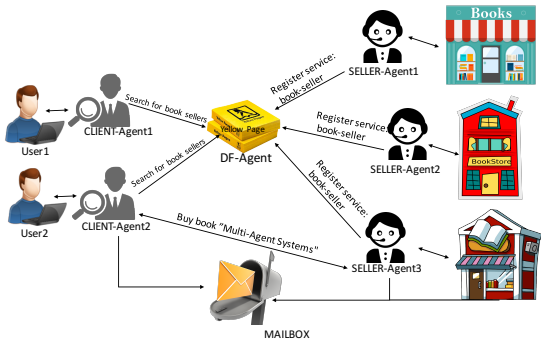


Figure 1: Scenario1: Overview of the general system architecture.

When a user creates a new selling agent, this agent registers itself in the Yellow Page by offering the service of book-seller. A selling agent manages a book catalog for a book store. Each user increments its own catalog at runtime by adding new books for sale. To add a book for sale, the user informs the name of the book and the price that he would like to receive for the book. A client agent is responsible for seeking and buying the book that a buyer user is looking for. Once created, the client agent is released into the marketplace, where it investigates which selling agents have the desired book and it buys the book from the seller that has the best price.

We also added a mailbox to the application. Our goal is to simulate interactions between agents that are different from ACL message communication. In such case, this interaction is performed by sharing a common resource among agents, that is, the mailbox. After selling the book, the seller agent sends a virtual copy of the book to the mailbox, while the client agent verifies if the book has been delivered. If the client agent buys a book and it does not find the book in the mailbox after a time, the client agent will fail.

3. TEST APPROACH: THE SOLUTION ARCHITECTURE

We developed a publish-subscribe based architecture as a foundation for generating different kinds of test applications for MASs. Our goal is to provide mechanisms that capture and process logs generated by agents automatically. As depicted in Figure 2, our architecture consists of three layers: MAS Application (L1), Publish-Subscribe Communication (L2), and Test Applications (L3). The Publish-Subscribe Communication layer uses the RabbitMQ platform [18] for delivering logs from agents (publishers) to be consumed

[1]The source of the test and the MAS application systems are available at http://www.inf.puc-rio.br/ nnascimento/MAS-tests.html

by test applications (subscribers). To understand more about the characteristics of RabbitMQ that we used in our approach, see `https://www.rabbitmq.com/tutorials/tutorial-five-java.html` (Accessed in 03/2017).

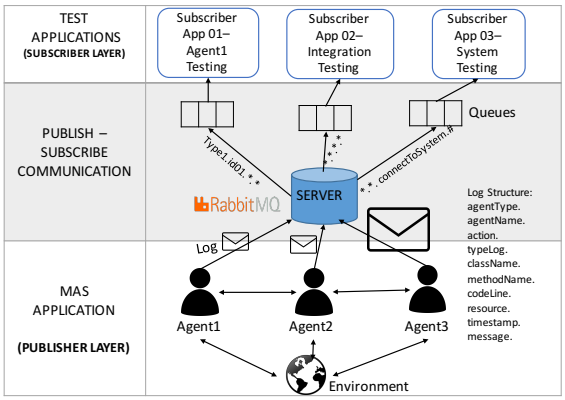


Figure 2: A Publish-Subscribe-based architecture to test MASs.

Each agent publishes logs with annotations that are composed of the following tags:

- *agentType*: the type of the agent (e.g CLIENT, SELLER, VEHICLE). In JADE, it refers to the name of the container where this agent lives;
- *agentName*: the name provided for the agent by the system developer/user (e.g client01, client02, seller01);
- *action*: the event that caused the log generation (e.g connectToSystem, searchBookInCatalogue, beDestroyed);
- *typeLog*: types of logs (e.g error, info, warning);
- *className*, *methodName*, *codeLine*: necessary information to identify the part of the code that generated the event;
- *resource*: the main resource that has been manipulated or requested by an agent during an event execution (e.g book1, book3, memory). It may be used to investigate all events that are related to a specific resource;
- *timestamp*: time that the log was created. Used to sort all events into a single timeline [16];
- *message*: a description of the event.

Thus, a log message must meet the pattern "(agentType).(agentName).(action).(typeLog).(className).(methodName).(codeLine).(resource).(timestamp).(message)." Each agent-based application must specify a set of values that can be used to fill these tag fields.

As depicted in Figure 3, all agents in the MAS application layer are also a TestableAgent type. A Testable agent uses RabbitMQ properties to send logs with annotations as messages. These logs can be published from any part of an agent's code. Some tags fields are automatically filled in by the TestableAgent class and JADE properties, such as agentType, agentName and timestamp.

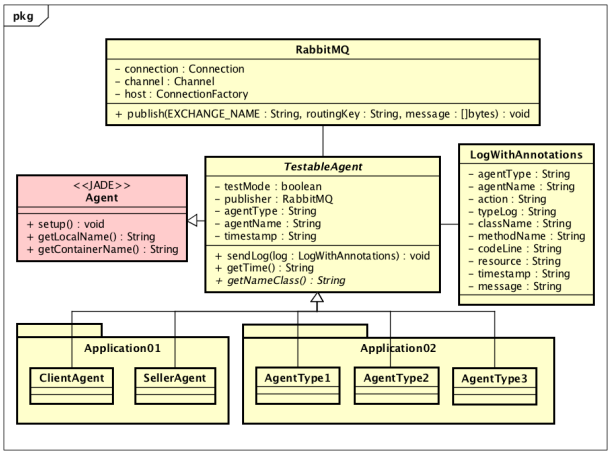


Figure 3: Simplified class diagram for creating testable MASs.

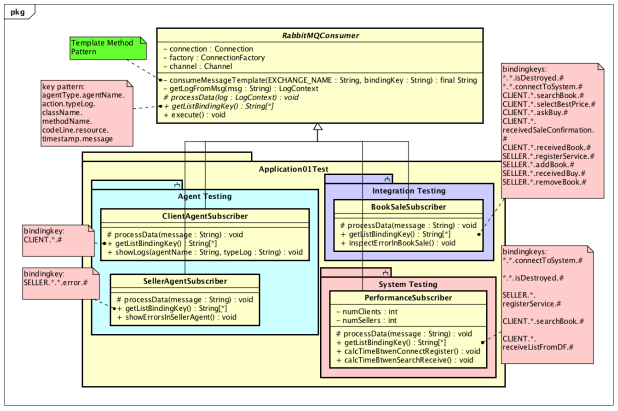


Figure 4: Simplified class diagram for creating applications for testing a MAS application at agent, integration and system levels.

The RabbitMQ autonomously delivers log messages to queues according to their tags' values. As shown in Figure 4, each test application defines a binding key in order to subscribe itself to consume messages from a specific queue. For example, a test application that monitors only error logs from SELLER agents must have the binding key "SELLER.*.*.error.#." Therefore, this application will

consume any log with the tuples (agentType,SELLER) and (typeLog,error). It is also possible to create applications that use multiple bindings. For example, if a performance test application needs to calculate the number of SELLER and CLIENT agents that are connected to the system, this application will have to consume logs with different action values. Thus, it needs to consume logs with the tuples (action,connectToSystem) and (action,beDestroyed).

Test applications do not interfere on the execution of each other. Each test class extends the class RabbitMQConsumer that starts an independent process to consume messages from a specific queue. We used the Template Method Pattern [19] to model the consumeMessage method. Thus, to consume and process particular log messages, a test class must overwrite and customize the methods getListBindingKey() and processData().

4. TESTS AND RESULTS

By using our proposed architecture, we created some test applications to execute functional tests at agent and group levels. Thus, this section presents part of the test plan that we created and performed for testing the application presented in the section 2. We provide the complete list of the functional and performance tests that we executed in [20].

4.1. Agent and Group Tests

We executed various test cases, taking eight parameters into account: (i) level (i.e. agent or group); (ii) function (e.g it is composed of a set of actions, for example, the function buy book may be composed of askBuy and soldBook actions); (iii) procedure (e.g a general description of the test); (iv) input (i.e a resource, a component); (v) expected value (e.g the result that will be produced when executing the test if the program satisfies its intended behavior); and (vi) validation method (e.g strategies that a tester performs to evaluate the system, comparing the program execution against expected results). Each test case execution produced several logs with meta-information annotations, which were consumed by test applications. Then, we used these logs as a validation method, as shown in table 1.

To validate a test case, the test application must verify if logs are appearing in the order described in the Validation Method column. Therefore, after the developer informs the logs from the validation column, the test application will automatically create a state machine, where each state represents an action. For example, Figure 5 illustrates the state machine that was created to validate the execution of the "buy Book" function. As shown, the verification program defines the transition between states as a log. A transition will only occur when the expected log appears. Each state has a maximum wait time for the expected log. Thus, if the maximum wait time exceeds, an error linked to the current state will be generated. This situation indicates that an agent performed an unexpected behavior and the action was not successful executed.

Table 1: Functional tests at agent and integration (group) levels (Simplified Table).

Level	Func.	Procedure	Input	Expected Value	Validation Method (Logs sorted into a timeline)
Agent	create Selling Agent	User creates a new agent	1.Type: SELLER 2. Name: seller1	seller1 agent is registered as book-seller	1)SELLER.seller1. connectToSystem. info.# 2)SELLER.seller1. registerService.info.#
	add Book	User adds a new book to seller1	1.Book's name: book1 2.Book's price:10	book1 is in seller1's catalogue	1)SELLER.seller1. addBook.info.*.*. .*.*.'name:book1 and price:10'
Group	buy Book	client1 asks seller1 to buy book1	1.client1 2.seller1 3.book1	client1 receives book1 after two seconds	1)CLIENT.client1. askBuy.info.# 2)SELLER.seller1. receivedBuy.# 3)SELLER.seller1. removedBook.# 4)SELLER.seller1. soldBook.# 5)CLIENT.client1. receivedSaleConf.# 6)CLIENT.client1. receivedBook.#

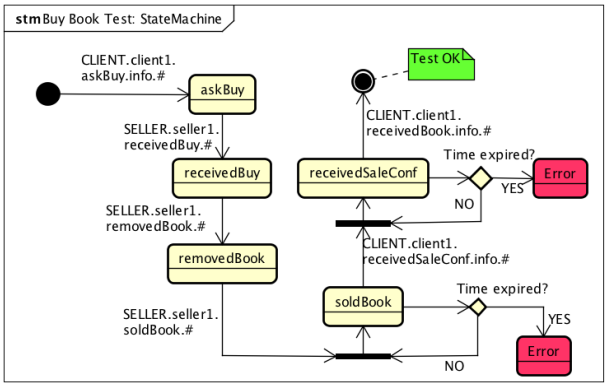


Figure 5: Simplified state machine for verifying test cases generated for the function buy book.

In order to force test failure and verify if these test applications were able to identify faults, we forced certain classes to act incorrectly during the execution of the program over some integration tests. For example, to test the function "buy Book", we inserted a defect that makes a Seller agent to die after accepting a book sale. Therefore, a client agent that bought a book from this seller, did not find this book in the mailbox and failed. As the test application did not receive the log "CLIENT.client1.receivedBook.info.#", its state machine indicated a failure in the state "receivedSaleConf." Figures 6 and 7 depict the logs that were generated by agents while this situation was executing.

```
[x] Sent 'SELLER.seller1.create.INFO.SellerAgent.initAgent.40.agent.2017.3.5.22.30.18.41.
[x] Sent 'SELLER.seller1.connectToSystem.INFO.SellerAgent.setup.56.agent.2017.3.5.22.30.18.47.
[x] Sent 'SELLER.seller1.createCatalogue.INFO.SellerAgent.setup.55.catalogue.2017.3.5.22.30.18.299.
[x] Sent 'SELLER.seller1.registerService.INFO.SellerAgent.registerService.71.yellowpage.2017.3.5.22.30.18.
[x] Sent 'SELLER.seller1.addBook.INFO.SellerAgent.addBookToCatalogue.109.book.2017.3.5.22.30.18.541.book1:
[x] Sent 'SELLER.seller1.receivedBudget.INFO.SellerAgent.receiveBudgetRequest.133.book.2017.3.5.22.31.58.7
[x] Sent 'SELLER.seller1.receivedBuy.INFO.SellerAgent.receiveMsgToBuy.171.book.2017.3.5.22.31.58.777.clien
[x] Sent 'SELLER.seller1.soldBook.INFO.SellerAgent.receiveMsgToBuy.171.book.2017.3.5.22.31.58.788.client:
[x] Sent 'SELLER.seller1.isDestroyed.WARNING.SellerAgent.takeDown.98.agent.2017.3.5.22.31.59.570.
[x] Sent 'SELLER.seller1.isDestroyed.ERROR.SellerAgent.takeDown.98.agent.2017.3.5.22.31.59.653.
```

Figure 6: Logs generated by Seller1.

```
[x] Sent 'CLIENT.client1.connectToSystem.INFO.ClientAgent.setup.41.agent.2017.3.5.22.30.58.358.
[x] Sent 'CLIENT.client1.create.INFO.ClientAgent.initAgent.54.agent.2017.3.5.22.30.58.358.
[x] Sent 'CLIENT.client1.searchBook.INFO.ClientAgent.setup.53.book.2017.3.5.22.31.58.661.book:
[x] Sent 'CLIENT.client1.receiveListFromDF.INFO.ClientAgent.setup.73.yellowpage.2017.3.5.22.31.
[x] Sent 'CLIENT.client1.askPrice.INFO.ClientAgent.buyBook.145.book.2017.3.5.22.31.58.703.ask b
[x] Sent 'CLIENT.client1.receivedBudget.INFO.ClientAgent.buyBook.163.agent.2017.3.5.22.31.58.75
[x] Sent 'CLIENT.client1.selectBestPrice.INFO.ClientAgent.buyBook.182.agent.2017.3.5.22.31.58.7
[x] Sent 'CLIENT.client1.askBuy.INFO.ClientAgent.buyBook.145.agent.2017.3.5.22.31.58.776.seller
[x] Sent 'CLIENT.client1.receivedSaleConfirmation.INFO.ClientAgent.buyBook.180.book.2017.3.5.22
[x] Sent 'CLIENT.client1.receivedBook.ERROR.ClientAgent.getBookInPostOffice.229.book.2017.3.5.2
```

Figure 7: Logs generated by Client1.

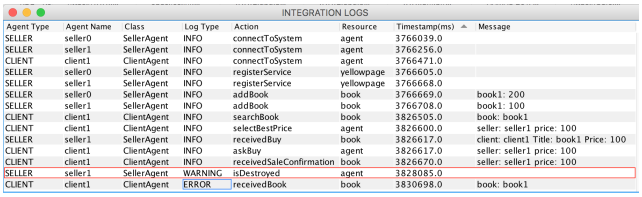
As each agent is running separately, to identify and understand what caused this failure, the tester would have to inspect the log file from each one of the agents. This work could be so difficult if the number of agents or the number of log messages was higher. Thus, by using our proposed solution, a test application can automatically select those logs from different agents that are probably to be essential for a specific test case and show them sorted in a single timeline.

In order to specify which characteristics need to be monitored from logs during the execution of a test cases set, the developer must establish a list of binding keys and override the method getListBindingKey() from the RabbitMQConsumer class (Figure 4). The list of binding keys will determine which test cases can be covered by the test application. For example, to cover the test cases that are listed in Table 1, it is necessary to establish binding keys that will make the test application able to receive all logs that are described in their "Validation Method" columns.

The code below shows the method getListBindingKey() that was used by a test application to cover these three test cases. If the developer wants this test application covering more test cases, he needs to add more binding keys to allow the test application to consume different logs.

```
@Override
public String [] getListBindingKey () {
BindingKey bd = new BindingKey();
String [] listKey = new String [11];
listKey [1] = bd.createBindingKey(LogValue.Action.connectToSystem);
listKey [2] = bd.createBindingKey(LogValue.AgentType.CLIENT, LogValue.Action.askBuy);
listKey [3] = bd.createBindingKey(LogValue.AgentType.CLIENT, LogValue.Action.receivedSaleConfirmation);
listKey [4] = bd.createBindingKey(LogValue.AgentType.CLIENT, LogValue.Action.receivedBook);
listKey [5] = bd.createBindingKey(LogValue.AgentType.SELLER, LogValue.Action.registerService);
listKey [6] = bd.createBindingKey(LogValue.AgentType.SELLER, LogValue.Action.addBook);
listKey [7] = bd.createBindingKey(LogValue.AgentType.SELLER, LogValue.Action.receivedBuy);
listKey [8] = bd.createBindingKey(LogValue.AgentType.SELLER, LogValue.Action.removedBook);
return listKey ;
}
```

As a result, the interface depicted in Figure 8 shows all logs that were consumed by a test application according to this binding key list. In addition, all logs are organized in a single timeline. As shown, not all logs depicted in Figures 6 and 7 were presented in this interface, but only the relevant logs to inspect the execution of these test cases. Thus, we were able to verify these logs in order to find the fault that generated the failure indicated by the state machine.



Agent Type	Agent Name	Class	Log Type	Action	Resource	Timestamp(ms)	Message
SELLER	seller0	SellerAgent	INFO	connectToSystem	agent	3766039.0	
SELLER	seller1	SellerAgent	INFO	connectToSystem	agent	3766256.0	
CLIENT	client1	ClientAgent	INFO	connectToSystem	agent	3766471.0	
SELLER	seller0	SellerAgent	INFO	registerService	yellowpage	3766605.0	
SELLER	seller1	SellerAgent	INFO	registerService	yellowpage	3766668.0	
SELLER	seller0	SellerAgent	INFO	addBook	book	3766669.0	book1: 200
SELLER	seller1	SellerAgent	INFO	addBook	book	3766708.0	book1: 100
CLIENT	client1	ClientAgent	INFO	searchBook	book	3826505.0	book: book1
CLIENT	client1	ClientAgent	INFO	selectBestPrice	agent	3826600.0	seller: seller1 price: 100
SELLER	seller1	SellerAgent	INFO	receivedBuy	book	3826617.0	client: client1 Title: book1 Price: 100
CLIENT	client1	ClientAgent	INFO	askBuy	agent	3826617.0	seller: seller1 price: 100
CLIENT	client1	ClientAgent	INFO	receivedSaleConfirmation	book	3826670.0	seller: seller1 price: 100
SELLER	seller1	SellerAgent	WARNING	isDestroyed	agent	3828085.0	
CLIENT	client1	ClientAgent	ERROR	receivedBook	book	3830698.0	book: book1

Figure 8: Application View - Agent and Integration tests. Fault detection in test case execution.

Test Results

As shown in Table 1, we executed some functional tests at agent and group levels. By using state machines, the test applications were able to validate these test cases by comparing the logs consumed from the MAS publisher against the logs listed in the "Validation Method" column. In addition, we also conducted some tests by inserting software failures and verifying if our test software could be useful for detecting faults. As a result, after the state machine had indicated a failure, the developer could use the interface to identify the fault and reduce the time of diagnosis.

5. Conclusions and Open Challenges

We believe these results are promising. We presented a decoupled architecture that allows a developer to execute tests simultaneously and independently while running a MAS. In addition, we provided evidence of the usability of our proposal, using it to test a known MAS application. We showed that it is possible to develop different tests for a multiagent system at different levels by using logs containing meta-information annotations and a publish-subscribe technology.

5.1. Testing Non-Deterministic Applications

However, MASs usually are much more complex than the experiment that was used in this work. Current approaches modeled by using a MAS may involve non-deterministic characteristics that were not addressed by this paper, such as *learning* [21], *self-adaptation* and *self-organization* (SASO) [22]. In fact, there is a gap in the literature regarding the test of systems with these features. There are few approaches to inspect the emergence process in a self-organizing MAS system [23] [24], and all of them do MAS design based only on simulation techniques. One reason is the difficulty of specifying expected results for non-deterministic applications, especially in actual environments. Nonetheless, we believe our approach opens the way for more experiments in testing Multiagent Systems, since it provides ways for testing a MAS at different levels. For example, as a self-organizing MAS system enables the emergence of social features based on the behavior of individual

agents, to test this kind of system it is necessary to perform tests at single and group levels.

5.2. Deploying Agents in a Distributed Environment

For instance, all agents are running on a single machine and the proposed approach assumes that there is a central clock so that the timestamp in each message can be used to sort events and measure throughput. If the tester needs to evaluate the MAS application in a distributed environment, he can use RabbitMQ to create a common publisher that publishes logs from agents localized on different machines. However, new challenges will arise. In the general case of distributed agents, possibly running on machines in different continents, a synchronization mechanism is required.

5.3. Predictive Analysis

For each application, there is a set of predetermined values that can be used in tags. Thus, we can codify and normalize these values to use them as inputs of a temporal neural network, which is a known structure of predictive analysis. By consuming temporal logs, a test application may use a temporal neural network to process log information in order to predict errors.

Acknowledgements This work has been supported by the Laboratory of Software Engineering (LES) at PUC-Rio. Our thanks to CNPq, CAPES, FAPERJ and PUC-Rio for their support through scholarships and fellowships.

References

[1] C. Lucena, *Software engineering for multi-agent systems II: research issues and practical applications*. Springer Science & Business Media, 2004, vol. 2.

[2] M. Pěchouček and V. Mařík, "Industrial deployment of multi-agent technologies: review and selected case studies," *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 3, pp. 397–431, 2008.

[3] P. Ammann and J. Offutt, *Introduction to software testing*. Cambridge University Press, 2008.

[4] 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.

[5] E. Serrano, A. Muñoz, and J. Botia, "An approach to debug interactions in multi-agent system software tests," *Information Sciences*, vol. 205, pp. 38–57, 2012.

[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] R. Coelho, E. Cirilo, U. Kulesza, A. von Staa, A. Rashid, and C. Lucena, "Jat: A test automation framework for multi-agent systems," in *2007 IEEE International Conference on Software Maintenance*. IEEE, 2007, pp. 425–434.

[8] 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 2015 International Conference on Autonomous Agents and Multiagent Systems*, 2015, pp. 1071–1079.

[9] F. Cunha, A. D. da Costa, M. Viana, and C. J. P. de Lucena, "Jat4bdi: An aspect-based approach for testing bdi agents," in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2015 IEEE/WIC/ACM International Conference on*, vol. 2. IEEE, 2015, pp. 186–189.

[10] V. J. Koeman, K. V. Hindriks, and C. M. Jonker, "Automating failure detection in cognitive agent programs," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 2016, pp. 1237–1246.

[11] 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.

[12] A. Ferrando, D. Ancona, and V. Mascardi, "Decentralizing mas monitoring with decamon," *Autonomous Agents and Multi-Agent Systems*, vol. 16, 2017.

[13] J. Botía, A. Lopez-Acosta, and A. Skarmeta, "Aclanalyser: A tool for debugging multi-agent systems," 2004.

[14] F. Bellifemine, G. Caire, T. Trucco, G. Rimassa, and R. Mungenast, "Jade administrator's guide," *TILab (February 2006)*, 2003.

[15] G. Malkomes, K. Lu, B. Hoffman, R. Garnett, B. Moseley, and R. Mann, "Cooperative set function optimization without communication or coordination," in *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems*, 2017.

[16] T. P. de Araújo and A. von Staa, "Supporting failure diagnosis with logs containing meta-information annotations," *Technical Reports in Computer Science. PUC-Rio. ISSN 0103-9741*, vol. 14, p. 21, 2014.

[17] A. Gupta, O. D. Sahin, D. Agrawal, and A. E. Abbadi, "Meghdoot: content-based publish/subscribe over p2p networks," in *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*. Springer-Verlag New York, Inc., 2004, pp. 254–273.

[18] RabbitMQ, "Rabbitmq," Available in https://www.rabbitmq.com/, 10 2016.

[19] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design patterns: Abstraction and reuse of object-oriented design," in *European Conference on Object-Oriented Programming*. Springer, 1993, pp. 406–431.

[20] N. M. do Nascimento, C. J. M. Viana, A. v. Staa, and C. J. P. de Lucena, "A publish-subscribe based approach for testing multi-agent systems," *Technical Reports in Computer Science. PUC-Rio. ISSN 0103-9741*, vol. 2016, p. 21, 2016.

[21] J.-P. Briot, N. M. de Nascimento, and C. J. P. de Lucena, "A multi-agent architecture for quantified fruits: Design and experience," in *28th International Conference on Software Engineering & Knowledge Engineering (SEKE'2016)*. SEKE/Knowledge Systems Institute, PA, USA, 2016.

[22] N. M. do Nascimento and C. J. P. de Lucena, "Fiot: An agent-based framework for self-adaptive and self-organizing applications based on the internet of things," *Information Sciences*, vol. 378, pp. 161–176, 2017.

[23] L. Gardelli, M. Viroli, and A. Omicini, "On the role of simulation in the engineering of self-organising systems: Detecting abnormal behaviour in mas," 2005.

[24] C. Bernon, M.-P. Gleizes, and G. Picard, "Enhancing self-organising emergent systems design with simulation," in *International Workshop on Engineering Societies in the Agents World*. Springer, 2006, pp. 284–299.

Practical Reasoning in an Argumentation-based Decision BDI Agent: a Case Study for Participatory Management of Protected Areas

Pedro Elkind Velmovitsky[1], Jean Pierre Briot[2, 1], Marx Viana[1], Carlos Lucena[1]
[1] Laboratório de Engenharia de Software (LES), Pontifícia Universidade Católica (PUC-Rio), Rio de Janeiro, Brazil
[2] Sorbonne Universités, UPMC Univ Paris 06, CNRS, Laboratoire d'Informatique de Paris 6 (LIP6), Paris, France
{pvelmovitsky, mleles, lucena}@inf.puc-rio.br, jean-pierre.briot@lip6.fr, alessandro.sordoni@gmail.com

***Abstract*— This paper describes the implementation of an argumentation system used for participatory management of environmental protected areas, more precisely to model the decision of a park manager artificial agent. This implementation is based on a BDI agent architecture, namely the Jason/*AgentSpeak* framework/language. After introducing the principles of BDI architecture and of argumentation systems, we will detail how we model arguments within the BDI (Belief-Desire-Intention) architecture. Then, we present the argumentation-based model of deliberation and decision by the park manager agent as a case study. We show how our argument-based approach allows to model various cognitive profiles of park managers (more conservationist or more sensitive to social issues), through different knowledge bases. We show examples of decisions produced by the park manager agent and examples of traces of arguments used during deliberation, which could be a base for explaining decisions. Before concluding, we point out future directions, such as using argumentation as a basis for negotiation between various agents.**

Keywords. Agent architecture; BDI architecture; Argumentation; Decision; Participatory management.

I. INTRODUCTION

The general context of this work is an ongoing research project exploring computer support for *participatory management* of protected areas. We have designed a serious game (more precisely a role-playing game), inspired by real management of national parks in Brazil [2]. The objective is to train people about participatory management of protected areas, allowing players to explore *negotiation strategies* in order to address conflicting views: in these protected areas, the stakeholders (such as environmentalist, tourism operator, traditional population representative, etc.), discuss, negotiate and take decisions about *environment management*. In practice, these decisions are about the *type* (level) of *conservation* for each sub-part of the park. Example among predefined types are *Intangible* (full conservation) and *Extensive* (flexible indirect use of resources). A special role is the *park manager*, who acts as an arbitrator in the game, making a final decision about the types of conservation, justifying (or, at least, *explaining*) its decision to the stakeholders.

Our research project [2] explores the use of various advanced computer techniques as a support for *assistance* to the players. One of the approaches explored is the use of *argumentation* systems, as a support for *decision* and for *negotiation*, and last but not the least, for *explanation*. We use this approach to model the decision of the park manager, through an internal deliberation between arguments [5]. We may also model the decisions of other players (roles) and support automated negotiation between them (human or artificial, as for instance in [8]).

In this paper, we will describe the modeling and implementation of an artificial agent playing the role of the park manager, based on an argumentation system (based on [3]), and modeled and implemented in a *BDI agent architecture*. More precisely, we are using the *Jason* framework, an implementation of the *AgentSpeak* agent programming language, based on the BDI architecture [4]. Said another way, we are representing arguments and their management in *AgentSpeak*. The first version of the implementation has already been completed and tested. We will report on our experience as well as pointing out future directions. The reminder of this paper is organized as follows: Section II discusses related work, while section III focuses on the background for BDI and for the Jason platform. Section IV presents the use of argumentation theory in BDI, Section IV presents the modeling and implementation of a BDI agent using argumentation and Section VI results. Finally, Section VII presents our conclusions and future work.

II. RELATED WORK

JogoMan-ViP [8] is a distributed role-playing game similar in spirit to our SimParc project, as its domain is about participatory management of hydric resources. It has introduced artificial players, also implemented through *AgentSpeak*/BDI on top of the Jason framework. This pioneering work has been an important influence. That said, the model of decision of artificial players is relatively simplified and with a predefined and fixed protocol for negotiation (to facilitate the interface between artificial players and human players). Our objective in using argumentation as the basis for decision, explanation and negotiation is even more ambitious.

DOI reference number: 10.18293/SEKE2017-153

Simulación by [12] experimented with the use of artificial agents as assistants of human players, as an innovative and proactive type of interface. Assistant agents can make suggestions to human players, based on the model of a good strategy for the game combined with a learning mechanism. This work has also been an inspiration for our project. But we want to explore various models of the game, including *predictive* models (to estimate viability and resilience) and inner deliberation and justification models (by using *argumentation*).

Regarding argumentation systems, the theoretical framework by Rahwan and Amgoud [3] is interesting and a major source of inspiration, in that it is one of the first to use argumentation not only for *epistemic* reasoning (i.e., to create or modify knowledge) but also to *practical* reasoning (i.e., for reasoning about what to do and also how to do it). [13] is an implementation of an argumentation system based in Prolog. [14] is a preliminary integration of an argumentation system into a BDI (*AgentSpeak*/Jason architecture), with similar objective to ours, but focusing on the implementation of defeasible logic (logical rules that can be refuted).

III. BDI AND JASON

There are many ways to model the behavior of agents such as the BDI (Belief-Desire-Intention) model. To talk about this model, according to [4], we need to address the idea that we can talk about computer programs as if they had a "*mental state*". Thus, when we talk about a BDI system, we are talking about computer programs with computational analogues of beliefs, desires and intentions.

Beliefs are the information the agent has about the environment. This information, however, is something the agent believes in but it may not be necessarily true. As an example, an agent may perceive from its environment the fact that it is raining. However, the rain may stop before the next reasoning cycle of the agent – in this case, his belief is outdated and incorrect. *Desires* are the possible states of affairs that the agent might like to accomplish. That does not mean, however, that the agent will act upon it – it is a potential influencer of the agent's actions. *Intentions* are the state of affairs that the agent has decided to act upon. In other words, intentions can be considered as a selected option between the potential set of options/desires that the agent has decided to pursue.

Jason is a Java-based interpreter for the *AgentSpeak* language [4], providing a platform for the implementation and development of agents. This language is based on the BDI architecture [15] and allows programmers to customize the agent's knowledge base following logic sentences. The agents also have *goals* that express the wishes the agent wants to accomplish. For instance*, !buy(car)* means that the agent has the goal of buying a car. Furthermore, *AgentSpeak* provides a way to program and customize *plans* for the agents. These plans represent courses of action that the agents will take in order to achieve its goals. The overall syntax for a plan is *triggering event <- context: body*, where the *triggering event* represent changes which the agent will act upon; the *context* is used to check the current situation so as to determine whether a particular plan among various alternative ones is likely to succeed in handling the event; the *body* of a plan is the course of action the agent will take in order to handle the event that triggered the plan [4].

IV. ARGUMENTATION IN BDI

There have been several theories which look at formalizing the reasoning of autonomous agents based on mental attitudes, such as beliefs, desires and intentions (BDI). One of the main characteristics of this type of reasoning is the resolution of *conflicts*, since the goals and attitudes available to the agent may not always be compatible. In addition, the information that the agent has may not always be consistent, or it may be true at one moment but incorrect on the next one [3] and may be different for *another* agent. *Argumentation* is a promising approach to deal with such considerations: it is a mono-agent as well as a multi-agent process, in which an agent may decide alone, or adhere to the opinion of another agent, depending on the strength and validity of arguments. Furthermore, agents reserve the right to *revisit* their *opinions* in light of new information.

The classical logic proves inadequate to model such behaviors. For example, to verify the property of monotony: *If Φ, Δ and Γ denote sets of formulas in a formal reasoning system, then the property of monotony is stated as following: If Φ :- Γ and $\Phi \subset \Delta$ then Δ :- Γ.* The key here is that the addition of new formulas at Φ can never call into question the truth value of Γ; this is called a *closed world*. The interest in non-monotonic logic appears when we try to capture the notion of everyday reasoning, where definite conclusions are obtained from incomplete information which can be proven wrong or false; this is called an *open world* and *non-monotonous reasoning*, where the addition of new formulas at Φ can call into question the truth of Γ.

However, this logical approach is limited to *epistemic reasoning* and does not modulate *practical reasoning*, limiting its use in agent architectures. To deal with this, a new approach has been developed: the *argument*. As opposed to a proof, an argument may be invalidated, and by comparing arguments it is possible to manage inconsistencies in the agent's belief base. In order to formalize this notion, we refer to concepts used in [3] and [5]: *Let the classical deduction be denoted by :- and the logical equivalence $\equiv$. Then, an argument is a pair ⟨H, h⟩, such that 1) H is consistent; 2) H :- h; 3) H is minimal, that is, there is no subset of H which verifies 1 and 2.* From this definition, the authors define the attack relations *refute* and *block* between arguments: *Let ⟨H1, h1⟩ and ⟨H2, h2⟩ two arguments. ⟨H1, h1⟩* ***refute*** *⟨H2, h2⟩ iff $h1 \equiv \neg h2$; ⟨H1, h1⟩* block *⟨H2, h2⟩ iff $\exists h \in H2, h \equiv \neg h1$.*

V. MODELING AND IMPLEMENTATING MANAGER AGENT

To model the park manager, or other players in the game, along their respective roles, we used the formalism above. So, let us note *D* the set of desires, *B* the set of beliefs and *A* the set of actions. Let us suppose, for example, that: *B = {road, tourism_flow, beach}* and *A = {extensive_use, intangible_use}*. Each agent, then, will have the following rules and bases:

1) The *rules to generate desires* (later on, named *desires rules)* RD_i have the form: ϕ -: $\beta_1, ..., \beta_m, \phi_1, ..., \phi_n, \beta_i \in B$ *and* $\phi_i \in D$. If the agent has beliefs $\beta_1, ..., \beta_m$ and desires $\phi_1, ..., \phi_n$ then desire ϕ is satisfied. These rules belong to the base $BD = \{(RD_i, w_i)\}$, where RD_i represents a desire rule and w_i represents

the intensity (weight) of the conclusion of the desire ϕ. An example is: *+road : tourism_flow & beach <-+raise_tourism(3),* where *(3)* means that *Intensity* $_{(raise\_tourism)} = 3$;

2) The *decision rules RAj* have the form ϕ *-:* α, where $\alpha \in A$ and $\phi \in D$. If the agent takes the decision of performing action α, then desire ϕ is satisfied. These rules belong to the base $BA = \{(RAj, u_j\}$, where *RAj* represents a rule of desire and u_j represents the utility of the action α according to the desire ϕ. An example is: *+extensive_use(0.75) -> +raise_tourism(0.75)*, with $Utility_{raise\_tourism}$ *(extensive_use) = 0.75*.

It is important to note that, for an agent to decide which action to choose, he must compare the gain of each action. These gains are defined as $gain_{\phi}(\alpha) = intensity(\phi) * u_{\phi}(\alpha)$. (see more details in [5]).

TABLE I. DESIRE RULES (RD BASE) AND DECISION RULES (RA BASE)

Name	Rule
RD_1	*+road : tourism_flow & beach <-+raise_tourism(3)*
RD_2	*+waterfall : true <- +~raise_tourism(2)*
RD_3	*+forest : true <- + protect_forest(3)*
RD_4	*+forest_fire: true <- +prevent_fire(3)*
RD_5	*+beach : ~protect_forest(A) <-+protect_forest_argument(C)*
RA_1	*+extensive_use(0.75) : true <- !raise_tourism_ extensive_ use (0.75)*
RA_2	*+intangible_use(1.0):true<-!protect_forest_intangible_use(1.0); !prevent_fire_intangible_use(1.0/2); !protect_forest_prevent_fire_intangible_use(1.0)*

Suppose that the available actions are *extensive_use* and *intangible_use,* and the agent has the following beliefs about the area: *{"road","waterfall", "forest", "beach", "tourism_ flow", "forest_ fire"}*. Then, desire rules (*RD* base) and decision rules (RA base) are modeled as shown in Table I. Table II shows the *Plans* that are executed when the actions *extensive_use* and *intangible_use* become available. The *raise_tourism* desire related to *extensive_use* has utility *0.75*, the *protect_forest* and *prevent_fire* desires related to *intangible_us*e have utilities *1.0* and *0.5*, respectively.

It is important to note that the values of *intensity* and *utility* are modeled by the programmer, according to the manager's "personality". If this particular park manager is more *socioconservationist*, he may prefer a more *extensive use* of the area and therefore the utility from the desire *raise_tourism* may be bigger. The same applies if the park manager is more *preservationist* and therefore prefers a more *intangible use* of area.

With the rules and parameters mentioned above, there are two attack relations between explanatory arguments: refute attack between RD1 and RD2, since RD1's conclusion is *raise_tourism*, and RD2's conclusion is *~raise_tourism*; block attack between RD3 and RD5, since RD3's conclusion *protect_forest* is present in the body of RD5.

The park manager agent BDI, with its respective *RD* and *RA bases*, has been implemented utilizing the Jason plug-in for *Eclipse*. Then, an implementation of the mechanism to compare arguments in order to eliminate conflicts has been modeled in this architecture. For instance, in Table II, *!raise _tourism _extensive _ use (T)* is used to check if the *raise_tourism* desire was added from the *RD* in P_1 and P_2: since the agent has a rule that adds the *raise_tourism* desire, P_1 will be executed. If the agent's belief base did not include a *road*, for instance, then the *raise_tourism* desire would not be added from RD_1, and the agent would not believe that this desire is feasible – that is what the *not raise_tourism(A)* clause means. In this case, P_2 would be executed. The goals *!protect_forest_intangible_use(T)* and *!prevent_ fire_ intangible_ use(T)* are used in a similar fashion, and are not described here.

The *protect_forest_argument(C)* goal is used to check if a block attack relation exists in P_3 and P_4: in P_3, if the intensity of the negation of the *protect_forest desire*, denoted by *C*, is bigger than the intensity of the desire itself, denoted by *A*, then the *protect_forest* desire is removed. In P_4, if *C* is smaller than *A*, then the desire is not removed. In this implementation, *C* has been set to *4* and *A* to *3*, so P_3 applies and the desire is removed. The refute attack relation, on the other hand, is tested in P_5 and P_6. In P_5, if the intensity of the negation of the *raise_tourism* desire, denoted by *A*, is bigger than the intensity of the desire itself, denoted by *B*, then the *raise_tourism* desire is removed. In P_6, if A is smaller than *B*, then the desire is not removed. In this implementation, *A* has been set to *2* and *B* to *3*. So, P_6 applies and the desire is not removed. Lastly, the plan *!protect_forest_prevent_fire_intangible_use(T)* is used to consolidate all results. Since this goal is part of the body of the plan of the *intangible_use* action, its context needs to test if the desires *protect_forest, prevent_fire* and *raise_tourism* are available and if the action *extensive_use* is available. Then, it checks the gains of each action accordingly and selects an action to perform. For instance, P_7 and P_8 test if the desire *protect_forest* is present in the belief base and if the other desires are still feasible, if the *extensive_use* action is still available and if the gains of the *intangible_use* action are bigger than the gains of the *extensive_use* action. Similar clauses, omitted here, have been added for all possible combinations in the context, so that the agent will always know how to calculate the gains and choose an action. More details about the *AgentSpeak* language can be found in [4].

VI. RESULTS

The results of the implementation in Jason, as logged by the agent, are: 1) *extensive_use* action added from desire *raise_tourism*, with utility *0.75* and total gain *2.25*; 2) *raise_tourism* negation added with intensity *2* and desire *raise_tourism* with intensity *3* has bigger force; 3) *protect_forest* negation added with intensity *4* and desire *protect_forest* with intensity *3* removed; 4) *intangible_use* action added from desire *prevent_fire* with utility *0.5* and total gain *1.5*; 5) *extensive_use* action executed with total gain *2.25*, because *2.25* is bigger than *1.5*. Thus, the resulting decision by the park manager is *extensive_use.*

Suppose that we want to change the profile of the park manager from socioconservationist to preservationist. Then, the intensity of *~protect_forest* will be slightly adjusted from 4 to 2, causing the desire *protect_forest* to not be removed in the attack relation. The total utility of the intangible_use action will be 3*0.5 (*prevent_fire* desire) + 3*1.0 (*protect_forest* desire) = 4.5, surpassing the utility of the *extensive_use* action. The result, as traced by the agent, would be as following: 1) *extensive_use* action added from desire *raise_ tourism*, with utility *0.75* and total gain *2.25*; 2) *raise_tourism* negation

added with intensity *2* and desire *raise_tourism* with intensity *3* has bigger force; 3*) protect_forest* negation added with intensity *2* and desire *protect_forest* has bigger force *3*; 4) *intangible_use* action added from desire *protect_forest* with utility *1* and total gain *3*; 5) *intangible_use* action added from desire *prevent_fire* with utility *0.5* and total gain *1.5*; 6) *intangible_use* action executed with total gain *4.5*, because 2.25 is bigger than 2.25. Thus, the resulting decision is *intangible_use*.

This example shows the flexibility of the BDI model that makes use of the argumentation: it allows for the agent to deliberate and decide while eliminating conflicts between the bases. In this case, the conflict between the *protect_forest* desire and its negation is resolved by comparing their respective forces.

TABLE II. PLANS

Name	Plan
P_1	*+!raise_tourism_extensive_use(T): raise_tourism(A) <- .print (" extensive_use action added from desire raise_tourism, with utility ",D, " and total gain ", D*T)*
P_2	*+!raise_tourism_extensive_use(T) : not raise_tourism(A) <- .print ("extensive_use action added from desire raise_tourism, with utility ",D, " and total gain ", 0)*
P_3	*+protect_forest_argument(C) : protect_forest(A) & C>A <- .print ("protect_forest negation added with intensity ",C, " and desire protect_forest with intensity ", A, " removed");- protect_forest(A)*
P_4	*+protect_forest_argument(C) : protect_forest(A) & C<A <- .print ("protect_forest negation added with intensity ",C, " and desire protect_forest has bigger force ", A)*
P_5	*+~raise_tourism(A): raise_tourism(B) & A>B <- .print("raise_tourism negation added with intensity ", A, " and desire raise_tourism with intensity ",B, " removed");- raise_tourism(B)*
P_6	*+~raise_tourism(A): raise_tourism(B) & A<B <- .print("raise_tourism negation added with intensity ", A, " and desire raise_tourism with intensity ",B, " has bigger force")*
P_7	*+!protect_forest_prevent_fire_intangible_use(A) : not protect_ forest(E) & prevent_fire(D) & raise_tourism(C) & extensive_use (B) & C*B<=(D*A/2) <- .print("intangible_use action executed with total gain: ",(D*A/2),", because ", (D*A/2), " is bigger than ", C*B)*
P_8	*+!protect_forest_prevent_fire_intangible_use(A) : not protect_ forest(E) & prevent_fire(D) & raise_tourism(C) & extensive_use (B) & C*B>(D*A/2) <- .print("extensive_use action executed with total gain: ",C*B,", because ", C*B , " is bigger than ", (D*A/2))*

VII. CONCLUSION AND FUTURE WORK

In this paper, we have presented a prototype architecture of an artificial agent able to make decisions by comparing arguments. The architecture is based on a BDI architecture (namely, the Jason/*AgentSpeak* framework/language). We have modeled an argumentation system through different knowledge base layers and the relation of attacks between arguments, as a basis to select best viable arguments. We have tested the architecture by modeling a park manager in a serious game for participatory management of protected areas. The park manager artificial agent makes decisions about conservation types by examining and reasoning (comparing) facts and arguments about the protected area situation and concerns. As a future work, already under progress, we plan to completely automate the management of attacks between arguments, in order to increase the agent's autonomy and reasoning capacity. In addition, being able to track the agent's reasoning to choose an argument over another enables the agent to provide the user feedback about what is happening in the system. Last, argumentation could also be used, not only for internal deliberation within a single agent, but also for negotiation between agents by exchanging and evaluating arguments. This leads to the prospects of artificial negotiating players.

REFERENCES

[1] Hocine, N.; Gouaich, A.. A survey of agent programming and adaptive serious games. RR-11013, 2011, pp. 8. http://hal-lirmm.ccsd.cnrs.fr/lirmm-00577722

[2] Briot, J.-P., Irving, M. D. A., Vasconcelos Filho, J. E., de Melo, G. M., Alvarez, I., Sordoni, A., de Lucena, C. J. P. Participatory Management of Protected Areas for Biodiversity Conservation and Social Inclusion. In D. Adamatti (Ed), Multi-Agent Based Simulations Applied to Biological and Environmental Systems, Advances in Computational Intelligence and Robotics (ACIR) Book Series, IGI-Global, pp. 295–332.

[3] Rahwan, I.; Amgoud, L. An argumentation based approach for practical reasoning. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems. ACM, 2006. p. 347-354.

[4] Bordini, R. H.; Hübner, J. F.; Wooldridge, M. Programming multi-agent systems in AgentSpeak using Jason. John Wiley & Sons, 2007.

[5] Sordoni, A., Briot, J.-P., Alvarez, I., Vasconcelos, J., Irving, M., Melo, G. Design of a participatory decision making agent architecture based on argumentation and influence function – Application to a serious game about biodiversity conservation, RAIRO – An International Journal on Operations Research, 44(4):269–284. 2010.

[6] Barreteau, O. *et al*. Our companion modelling approach. Journal of Artificial Societies and Social Simulation, v. 6, n. 1, 2003.

[7] Le Page, C. *et al.* Participatory agent-based simulation for renewable resource management: the role of the Cormas simulation platform to nurture a community of practice. Journal of Artificial Societies and Social Simulation, v. 15, n. 1, p. 10, 2012.

[8] Adamatti, D. F.; Sichman, J. S.; Coelho, H. An analysis of the insertion of virtual players in GMABS methodology using the Vip-Jogoman prototype. Journal of Artificial Societies and Social Simulation, v. 12, n. 3, p. 7, 2009.

[9] Barreteau, O. *et al*. Role-playing games for opening the black box of multi-agent systems: method and lessons of its application to Senegal River Valley irrigated systems. 2001.

[10] Dignum, F. *et al*. Games and agents: Designing intelligent gameplay. International Journal of Computer Games Technology, v. 2009, 2009.

[11] Wei, W.; Alvarez, I.; Martin, S. Sustainability Analysis: Viability Concepts to Consider Transient and Asymptotical Dynamics in Socio-Ecological Tourism-based Systems. Ecological Modelling, V. 251, P. 103-113, 2013.

[12] Guyot, P.; Honiden, S. Agent-Based Participatory Simulations: Merging Multi-Agent Systems and Role-Playing Games. Journal of Artificial Societies and Social Simulation, V. 9, N. 4, 2006.

[13] Kakas, A.; Moraitis, P. Argumentation based decision making for autonomous agents. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems. ACM, 2003. p. 883-890.

[14] Panisson, A. R. *et al.* An Approach for Argumentation-based Reasoning Using Defeasible Logic in Multi-Agent Programming Languages. In: 11th International Workshop on Argumentation in Multiagent Systems. 2014.

[15] Wooldridge, M.; Jennings, N. R.; Kinny, D. A methodology for agent-oriented analysis and design. In: Proceedings of the Third International Conference on Autonomous Agents Agents 99. 1999.

Working Towards a BDI-Agent based on Personality Traits to Improve Normative Conflicts Solution

Paulo Henrique Cardoso Alves, Marx Leles Viana and Carlos José Pereira de Lucena
Laboratory of Software Engineering (LES) - Department of Computer Science
Pontifical Catholic University of Rio de Janeiro – PUC – Rio, Rio de Janeiro, RJ - Brazil
{palves, mleles, lucena}@inf.puc-rio.br

Abstract— **Norms exist to avoid and solve conflicts, make agreements, reduce complexity, and in general to achieve a desirable social order. However, norms eventually can be conflicting — for example, when there is a norm that prohibits an agent to perform a particular action and another norm that obligates the same agent to perform the same action at the same period of time. The agent' decision about which norms to fulfill can be defined based on rewards, punishments and agent goals. Sometimes, this balance will not be enough to allow the agent to make the best decision. In this context, this paper introduces an approach that considers the agent's personality traits in order to improve the solving process of normative conflicts.**

Keywords. Solving Normative Conflicts, Normative Agents, Multi-Agent Systems.

I. INTRODUCTION

Multi-agent Systems (MASs) are societies in which these heterogeneous and individually designed entities (agents) work to accomplish common or independent goals [1]. In order to deal with autonomy and diversity of interests among the different members, such systems provide a set of norms, which are mechanisms used to restrict the behavior of agents by defining what actions the agents are obligated, permitted, or prohibited to encourage the fulfillment of the norm through rewards definition and discouragement of norm violation by pointing out the punishments [2].

Norms must be complied with by a set of agents and include normative goals that must be satisfied by the addressees. In addition, norms are not always applicable, and their activation depends on the background in which agents are situated. In some cases, norms suggest the existence of a set of sanctions to be imposed when agents fulfill, or violate, the normative goal.

The decision-making process about which norms will be fulfilled or violated might be defined based on the agent's goals, rewards and punishment analysis [1]. Since an agent's priority is the satisfaction of its own goals, before complying with norms, the agent must evaluate their positive and negative effects on its goals [3] without hurting the agent's autonomy. Both rewards and punishment are the means for the agents to know what might happen independently of the agent's decision to comply, or not, with the norms. However, norms sometimes may conflict or be inconsistent with one another [4]. For instance, different norms can, at the same time, prohibit and obligate a state that the agent wants to fulfill and the simple balance between goals, rewards and punishment might not be enough for the agent to make the best decision.

The abstract normative agent architecture developed by [3], has four main steps: (i) agent perception, (ii) norm adoption, (iii) norm deliberation, and (iv) norm compliance. Within the norm deliberation step, conflicting norms are verified and a set of these norms is added to the norm compliance set.

We changed the internal process of the norm deliberation step to deal with conflicting norms by adding the agent's personality traits. These characteristics will help the software agents to make a better decision involving personality traits - for example, sense of duty and spiritual endeavor. We will present a user scenario that shows how the agents deal with normative conflicts when personality traits are considered. This will illustrate the new deliberation process proposed in this paper.

In this context, we present an approach to build emotional BDI-agents, which considers also others agent's personality traits [5] and emotions [6] to improve the decision-making process in the solution of normative conflicts. This approach aims at providing new resources for the agent to deal with conflicting norms supported by personality traits. As such, more human characteristics can be considered to improve the deliberation process. We built a software framework based on this approach, which provides a set of hot-spots and frozen-spots that enables the implementation of emotional normative functions. By using these new functions, it is possible to build emotional agents that: (i) use personality traits to improve the solution among normative conflicts, (ii) implement the agent's behavior similar to the human's behavior, and (iii) evaluate the effects on its desires with respect to the fulfillment or violation of a norm.

The reminder of this paper is organized as follows: Section II focuses on the background, while Section III discusses related work. Section IV presents the emotional BDI approach to solve normative conflicts. Section V describes a case study applying the emotional approach. Finally, Section VI presents our conclusions and future work.

II. BACKGROUND

A. Norms

Norms are designed to regulate the behavior of the agent, and therefore, a norm definition should include the address of the agent being regulated [7]. However, norms are different from laws, and they cannot force agents to comply with them. Agents are autonomous entities, so norms may only suggest and present the expected behavior.

In this work, we used the norm representation described in [9], which is composed by the representation of the element *norm* – it contains many different properties: (i) Addressee, (ii) Activation, (iii) Expiration, (iv) Rewards, (v) Punishments, (vi) Deontic

DOI reference number: 10.18293/SEKE2017-160

Concept, and (vii) State. For example, the property *Addressee* is used to specify the agents or roles responsible for fulfilling the norm.

In order to better understand the definition of norms and their representation, a user scenario was developed.

B. Conflicting Norms

Norms eventually may conflict, i.e., an action may be simultaneously prohibited and permitted, or it may be inconsistent, i.e., when an action is simultaneously prohibited and obliged [4]. These conflicts and inconsistencies may be caused by a norm that prohibits an agent to perform a particular action while another norm requires the same agent to perform the same action at the same time. For example, Fig. 1 presents a scenario of conflicting norms - when a norm defines that the buyer agent cannot give back the product bought and at the same time another norm defines that the buyer agent can return the product bought before opening it.

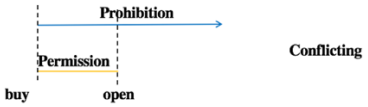


Figure 1. Conflict - Prohibition and Permission.

In short, conflicts may occur in different cases and situations, and dealing with them is extremely necessary to make the best decision.

C. BDI Archicteture

The BDI (*Belief-Desire-Intention*) model was proposed by [10] as a philosophical theory of practical reasoning, representing, respectively, the information, motivational and deliberative states of the agent. There are two main steps: (i) apply a filter to make a set of goals that the agent has to commit to base on his beliefs, and (ii) find a way to know how the desires produced can be fulfilled based on the available agent's resources [12].

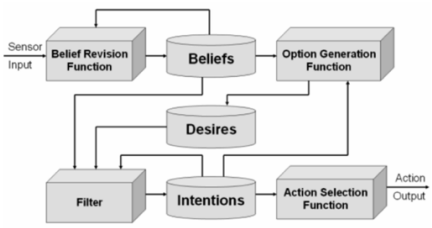


Figure 3. Generic BDI architecture [11].

The BDI model is composed by three mental states: (i) beliefs, which represent the environment factors that are updated after each action perceived — these beliefs represent the world knowledge; (ii) desires, which have information about the goals to be fulfilled — they represent the agent's motivational state, and (iii) intentions, which represent the action plan chosen. Fig. 3 shows these three mental states centralized and their interaction.

BDI architecture starts with a *Belief Revision Function* that makes a new belief set based on the agent's perception. Next, the *Option Generation Function* sets the agent's available options and desires, based on its own environment beliefs and intentions. The next function is a *Filter* that sets the agent's intentions based on its own beliefs, desires and intentions. Finally, the *Action Selection Function* sets the actions to be executed based on the current intentions.

Most BDI systems are inspired by the Rao e Georgeff [13] model. The authors presented an abstract BDI interpreter. This interpreter works with beliefs, goals and agent plans. As such, the goals are a set of concrete desires that may be evaluated altogether, avoiding a complex goal deliberation step. The interpreter's main activity is the means-end process achieved by plan selection and plan execution given a goal or event.

III. Related Work

This section describes some related work: (i) the solution for normative conflicts [1], [8], [15], [16]; (ii) architecture designs considering the agent's emotional state [14], and (iii) the agent's personality [5].

The authors in [5] built a decision process to work as part of the story-telling systems wherein narrative plots emerge from the acting characters behavior and personality traits. The process evaluates goals and plans to examine the plan commitment issue. The drives, attitudes and emotions play a major role in the process. However, the personality traits were not applied on MASs, which creates an opportunity to improve the agent's decision-making process to deal with normative conflicts.

Some approaches [1], [8], [15], [16] have been proposed in the literature to develop the agent that evaluates the effects of solving normative conflicts. For instance, the n-BDI architecture defined by Criado *et al.* [15] presents a model for building environments governed by norms. Basically, the architecture selects objectives to be performed based on the priority associated with each objective. An objective's priority is determined by the priority of the norms governing a specific objective. However, it is not clear in this approach how the properties of a norm can be evaluated. In addition, the approach does not support a strategy and neither consider the agent's personality traits to deal with conflicts between norms.

Lopes *et al.* [8] defined a set of strategies that can be adopted by agents to deal with norms as follows: *Pressured*, *Opportunistic* and *Selfish*. Although this work provides some mechanisms for the agents to collect norms, the authors provide a framework that can be extended to create simulations of normative multi-agent systems by including new strategies. In addition, this work can neither extend mechanisms to collect information during the simulations nor can extend mechanisms to generate norms and agent goals. Furthermore, the agent cannot detect and overcome normative conflicts.

Finally, Viana *et al.* [1] presents a modeling language and an architecture to build adaptive normative agents. The authors propose an approach to design and implement agents that are capable to adapt in order to deal with norms, detecting and overcoming normative conflicts. However, this research just measures norms contributions based on: (i) norm rewards and punishments; (ii) norm activation and expiration; (iii) deontic concept, and (iii) agent goals. As such, the agent can decide to fulfill or violate a norm. One item that was not broached by the authors is that they did not implement personality traits in their architecture to improve and overcome normative conflicts.

As none of this related work deals with norms conflicts using personality traits, this was the gap that we based on to propose our work. We aim at providing a better way to balance goals, rewards, punishment and personality traits to solve normative conflicts. To evaluate the norm contribution, we first use rewards and punishment values. With these values, we then continue to evaluate the norm contribution, now adding personality traits.

IV. EMOTIONAL BDI AGENTS: AN APPROACH

This section describes the main concepts required to understand the approach based on emotional BDI-agents used to improve the solution of normative conflicts. **Error! Reference source not found.**.

A. The Architecture

The emotional BDI-agents that can solve the normative conflicts approach were inspired on the concepts presented in the *background* and the related work section. We added both BDI features and personality traits in the normative deliberation process, mainly in relation to conflicts resolution. The architecture foundation was based on the abstract normative agent architecture developed in [3]. Fig. 4 presents our emotional BDI-agent architecture to solve normative conflicts.

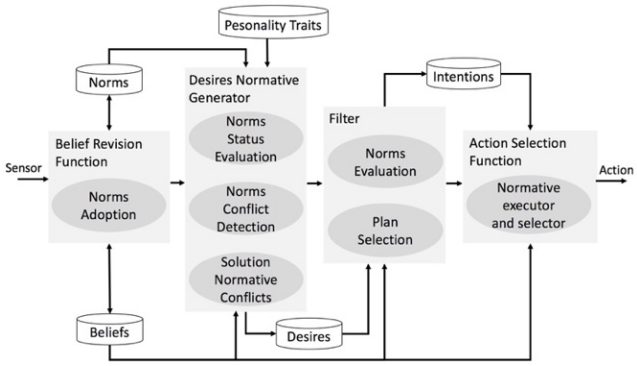


Figure 4. Internal architecture of the BDI-Agent based on Personality Traits to Improve Normative Conflicts Solution.

The most significant change was adding to the deliberation process a reasoning step that involves the BDI architecture and the personality traits approach. Both strategies work in a complementary way to make agents behavior more human, considering factors that were not used in the norms deliberation process in previous work. All of these changes refer only to the internal agent process. The decision-making process proposed has four steps, which is described below.

The first step involves the agent's perception in the *Belief Revision Function*, where the agent perceives the norms in the environment addressed to it by means of sensors. Then, the agent inserts into the *Norms* set the norms that it wants to fulfill by using the *Norms Adoption* function. After that, the agent updates its beliefs, taking into account these new norms.

The second step is the *Desire Normative Generator*, which is composed by three processes: (i) *Norm Status Evaluation* function, where the agent verifies which norms are activated or deactivated; (ii) *Norms Conflict Detection* function, where the agent verifies what the normative conflicts are, and (iii) *Solution Normative Conflicts* function, where the agent evaluates the norms contribution and solves the normative conflicts, also considering its personality traits. There are some personality traits considered: (i) Drives, such as Sense of duty and Spiritual endeavor, (ii) Attitudes, such as Careful and Adaptable, and (iii) Emotions, such as Anger and Fear as presented in [5]. As a result, a set of non-conflicting norms are exported to the next step. These norms are the agent's *Desires*.

The third step is the *Normative Filter*, which is composed by two processes: (i) *Norms Evaluation* function, where the agent evaluates the *Desires* set and it decides which norms will be fulfilled, and (ii) *Plan Selection* function, where the agent's best plans will be chosen in the *Intentions* set.

Finally, the fourth step is the *Action Selection Function*, which is composed by the *Normative executor and selector* function. This function receives the *Norms* set, which are the norms that the agent intends to fulfill. Last but not least, all of these steps help in the improvement of the normative conflict solving process, considering personality traits inserts into the BDI reasoning process.

B. The Framework

Inspired by the JSAN architecture [9], which uses different norms strategies to deal with norm, taking into account the different agent's social levels, as in [8]. We built a new approach by introducing personality traits aiming to improve the solution of the normative conflict.

The solving process of normative conflicts starts with the calculation of the norm's normative contribution, wherein for each norm the agent evaluates its rewards and punishments compared with the others norms addressed to it. Furthermore, we added a new step to improve this process, also taking into consideration the agent's goals and its personality traits. This new step consists in the evaluation of which normative goals can be fulfilled according to the agent's goals and its personality traits. The agent will verify which goals can be fulfilled based on its personality traits, so the agent uses its set of goals and analyze each conflicting norm, adding to the normative contribution an integer value to represent the compatibility between the agent's goals and the normative goals. The compatibility is defined by the evaluation of which of the agent's goals can be executed if a norm is fulfilled. As a result, some conflicting norms may have changed its normative contribution based on the use of the agent's personality traits. For instance, imagine the norm that obliges the agent to cross a damaged bridge. If the agent is careful (careful meaning the agent's personality trait) its normative contribution will be decreased because the agent does not have the intent to cross a damaged bridge — it is dangerous.

V. USER SCENARIO: GO HOME

As proof of concept, the user scenario "go home" will choose whether the agent goes home by bicycle, or by bus. The norms in this scenario are: (i) *Norm 1* prohibits the employee agent to go home by bicycle, and (ii) *Norm 2* obligates the employee agent to go home by bicycle. There are the *Norm 1* properties: (i) Name: Come back by Bus, (ii) Addressee: Employee, (iii) Deontic Concept: Prohibition, (iv) Reward: No health decrease, (v) Punishment: Be wet, (vi) Activation: It is raining, and (vii) Deactivation: It is sunny. *Norm 2* was defined by the following properties: (i) Name: Come back by Bicycle, (ii) Addressee: Employee, (iii) Deontic Concept: Obligation, (iv) Reward: Increase physical conditioning, (v) Punishment: Spend money with bus tickets, (vi) Activation: After work, and (vii) Deactivation: Be sick. Planning to go home, the employee agent checks the weather; if it is raining, it can go home by bus and as a consequence, it will violate *Norm 2*. However, if it is raining, but the employee agent has personality traits that induce its behavior to go home by bicycle, as a consequence, it will violate *Norm 1*. That is when the agent's internal process detects and tries to overcome the normative conflict between *Norm 1* and *Norm 2*.

Fig. 6 shows the normative conflict between *Norm 1* and *Norm 2* and our aim is to present improvement in the deliberation process to choose the norm that will be fulfilled, considering some characteristics, such as: (i) the rewards of the norm that will be fulfilled; (ii) the punishment of the norm that will be violated; (iii)

the agent's goals, and (iv) the personality traits — for instance, if the agent's goal is to increase physical conditioning, it will have adventurous spirit as a personality trait.

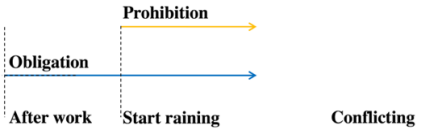


Figure 6. Go Home conflict area.

All of these attributes were mapped to integers values in our architecture to make possible the decision process to choose between *Norm 1 and Norm 2*. For comparison purposes, three different personality traits scenarios were developed for the employee agent: (i) adventurous spirit — *high* weight: (ii) adventurous spirit — *low* weight, and (iii) no personality trait.

We used the architecture proposed in this paper (see Fig. 4) in the first and second scenarios. For the three scenarios, we considered that the employee agent starts the *Norm Adoption* process to verify which norms are addressed to it. As a result, the employee agent perceives two norms. Note that these two norms are conflicting (see Fig. 6): both are active at the same time, and their deontic concept are opposite — obligation and prohibition. To choose which norms will be better fulfilled, the agent considers the normative contributions and its personality traits, except in the third scenario, where no personality trait is considered.

In the first scenario, we consider the employee agent with adventurous spirit — *high* weight to choose the norm that will be fulfilled in the conflict resolution process, also taking into account the norms punishments and rewards. The conflict resolution process measures, first of all, the norms rewards and punishments of each one of the conflicting norms, i.e., the agent verifies which goals can be executed, considering each one of the conflicting norms to be fulfilled. In sequence, the agent selects which norms will be fulfilled based on the agent's *Pressured* strategy. As a result, the employee agent will go home by bicycle. In the second scenario, we consider the employee agent with adventurous spirit — *low* weight. As a result, the employee agent decides that going home by bus will give it more benefits. This is because the agent has not enough motivation to fulfill its desires and as a consequence, it receives the punishments for not fulfilling the other norm.

Finally, in the third scenario, we consider the employee agent without personality traits, i.e., the agent always had the same behavior and considered only its own goals. We therefore assume that the BDI architecture with personality traits can change the agent's behavior, thus helping to improve the solution for the normative conflicts.

VI. Conclusion and Future Work

This paper proposes an approach to deal with conflicting norms by adding personality traits characteristics to the BDI architecture to improve the decision-making process that will decide which norms the agent shall fulfill. The main contributions of this research are: (i) include personality traits in the BDI architecture to improve the solving process of normative conflicts; (ii) implement different agent behaviors according to different personality traits, and (iii) make it possible to build software agents behaviors more similar to human behavior. As proof of concept, the approach presented in this paper can be verified by using the user scenario showed in Section V, where the agent needs to choose between bus or bicycle to go home once the weather conditions change. The emotional BDI-agent was able to reason about the norms it would like to fulfill, and to select the plans that met the agent's intention of fulfilling, or violating, the norms.

As future work, we are deciding on an experimental study in order to complete the evaluation of our approach. Furthermore, our aim is to study other BDI architectures and platforms to investigate the possibility of extending them to support the development of emotional BDI-agents to deal with norms and normative conflicts. Last but not least, we will extend our architecture to make it possible for the BDI-agent not only to use personality traits for the solving process of normative conflicts, but also for choosing the best plans that it can execute in order to deal with the norms addressed to it.

References

[1] Viana, M. L. ; Paulo Alencar ; Lucena, C. . A Modeling Language for Adaptive Normative Agents. In: EUMAS, 2016, Valencia. European Conference on Multi-Agent Systems, 2016.

[2] Figueiredo, K. Da S., Silva, V. T. Da, Braga, C. de O., Modeling norms in multi-agent systems with NormML. In Proceedings of the 6th international conference on Coordination, organizations, institutions, and norms in agent systems (COIN@AAMAS'10), Springer-Verlag, Berlin, Heidelberg, 39-57, 2010.

[3] López, L. F.; Márquez, A. A. An Architeture for Autonomous Normative Agents, IEEE, Puebla, México, 2004.

[4] Vasconcelos, W.W., Kollingbaum, M.J., Norman, T.J., Resolving conflict and inconsistency in norm-regulated virtual organizations. In Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems (AAMAS '07). ACM, New York, NY, USA, , Article 91 , 8 pages. 2007.

[5] Barbosa, S. D. J., Silva, F. A. G. Da, Furtado, A. L., Casanova M. A., Plot Generation with Character-Based Decisions. Comput. Entertain. 12, 3, Article 2 (February 2015).

[6] L. Padgham and G. Taylor, "A system for modelling agents having emotion and personality." in PRICAI Workshop on Intelligent Agent Systems, ser. Lecture Notes in Computer Science, L. Cavedon, A. S. Rao, and W. Wobcke, Eds., vol. 1209. Springer, 1996, pp. 59–71.

[7] Bordini, R. H.; Hübner, J. F.; Wooldridge, M. Programming Multi- Agent Systemns in AgentSpeak using Jason. [S.l.]: [s.n.], 2007.

[8] Lopez, Fabiola López. Social Power and Norms. Diss. University of Southampton, 2003.

[9] Viana, M. L. ; Paulo Alencar ; Everton Guimarães ; Cunha, F. J. P. ; Donald Cowan ; Lucena, C. . JSAN: A Framework to Implement Normative Agents. In: SEKE, 2015, Pittsburgh. The 27th International Conference on Software Engineering & Knowledge Engineering, 2015. p. 660-665.

[10] Bratman, Michael. Intention, plans, and practical reason. 1987.

[11] Wooldridge, Michael; Jennings, Nicholas R.; Kinny , David. A methodology for agent-oriented analysis and design. In: In Proceedings of the Third International Conference on Autonomous Agents Agents 99. 1999.

[12] Wooldridge, M. Ciancarini, P., Agent-Oriented Software Engineering: the State of the Art, LNCS 1957, Germany: Springer, p.45-58. 2001.

[13] Rao, Anand S., And Michael P. Georgeff. "BDI agents: From theory to practice." ICMAS. Vol. 95. 1995.

[14] D. Pereira, E. Oliveira, N. Moreira and L. Sarmento, "Towards an Architecture for Emotional BDI Agents," 2005 portuguese conference on artificial intelligence, Covilha, 2005, pp. 40-46.

[15] Criado, N., Argente, E., Noriega, P., and Botti, V. Towards a Normative BDI Architecture for Norm Compliance. COIN@ MALLOW2010, pages 65–81, 2010.

[16] dos Santos Neto, Baldoino F., Viviane Torres da Silva, and Carlos JP de Lucena. "Developing Goal-Oriented Normative Agents: The NBDI Architecture." International Conference on Agents and Artificial Intelligence. Springer Berlin Heidelberg, 2011.

A Virtual Environment for Problem-Based Learning in Software Engineering Education

Bruno Bessa
Centro de Informática
Federal University of Pernambuco
Recife, Brazil
brb@cin.ufpe.br

Simone C. dos Santos
Centro de Informática
Federal University of Pernambuco
Recife, Brazil
scs@cin.ufpe.br

***Abstract*— PBL (Problem-Based Learning) is a constructivist teaching method that uses real-life problems to develop skills and attitudes in students to solve them. This involves the inclusion of essential practices that are based on collaborative work and task management. As PBL is process-oriented, it is necessary to guide students in the definition of the solving process, ensure that students follow this process and monitor all the work, realizing improvements when necessary. In view of this, Virtual Learning Environments (VLE) can be very useful to support PBL processes in blended learning, providing a technological environment where the learning process can be managed. However, the literature shows that technological tools to support a PBL approach have only been explored to a limited extent and not effective. In this context, this paper proposes the "PBL-Coach", a VLE composed of a set of structured and measurable activities to support PBL in the Software Engineering Education. A case study was carried out with the aim of assessing the utility and usability of PBL-Coach in blended learning context, and the results showed a good acceptance level of its use.**

PBL; Learning Environment; Software Engineering Education

I. INTRODUCTION

New methods of teaching and learning have been adopted as a means of supporting the changes in the teaching and learning procedures that are aligned with the new requirements of the job market and a redefinition of the roles of those involved in education. For example, Problem-Based Learning (PBL), which can be defined as a method of teaching that is centered on the student, draws on real-life problems to trigger and encourage the learning of concepts by improving the skills and attitudes necessary to attempt to solve them. When contextualized in a realistic setting, the problems drive the students to seek ways of solving problems and acquire the necessary skills and attitudes to do this and thus be valued in their professional lives [1]. In this area, the recognized curricula in the area of Computing, such as the ACM/IEEE Computer Science Curricula, are being redefined in accordance with the inherent features of PBL, such as group learning, collaboration, problem-solving abilities, interpersonal communication and management skills.

In [2], the authors present a systematic mapping of the use of PBL in teaching Computing, describing the main features for an effective approach to PBL, together with its potential benefits and likely challenges. They stated that the benefits of PBL are closely linked to the methods and instruments of management, which allow the planning, execution, monitoring and continuous improvement of the solving-process conducted by students.

With regard to teaching Engineering Software in particular, the adoption of Information Communications Technology (ICT), such as Learning Management Systems (LMS) and Virtual Learning Environments (VLE), has not allowed teaching and learning to reach its full potential with regard to planning, monitoring and assessment [3]. In fact, many cases have a shortage of computer instruments and are restricted to limited ways of assisting the teaching and learning process, focused on collaborative work, but not include the definition of educational objectives, the linking between tasks and objectives, process to assess this work and its results. In [4], the authors make a comparison between a key list of VLE that are adopted in education. Among these, they underline the importance of the Open Simulator, Second Life, Active Worlds, Project Wonderland and Open Cobalt systems. On the basis of this analysis, it can be concluded that most of the systems are able to support online education, whether in terms of content or centered on the teacher, but there are only a few data that effectively show the support of PBL.

In [5], the author stresses that a VLE that is suited to PBL must be geared towards teamwork and be focused on a discourse that is concerned with the collaborative construction of knowledge. The author defines a particular term for this environment, "CMCPBL" (Computer-Mediated Collaboration for PBL). This refers to three significant features of this discourse and uses the work of [6] as a benchmark: (1) A focus on the background of the problems and an in-depth understanding of this problem; (2) Construction of open and collaborative knowledge; (3) Inclusion of all the participants in the learning process. In the light of these specific features, this work examines a set of activities based on the principles and features that arise from learning theories about PBL. These activities will be supported by a virtual learning environment with interactive technological resources, called PBL-Coach. The purpose of PBL-Coach is to support the implementation of the PBL method and ensure that it is adopted in compliance with its principles.

DOI: 10,18293 / SEKE2017-092

The proposed environment was validated by conducting a case study in the open education, which involved sixteen students and three teachers in a course on Agile Project Management on the period March-June, 2016. This study sought to answer two questions: "Q1) *What is the degree of usefulness of PBL-Coach in carrying out environmental activities based on the principles and main features of PBL*?"; "Q2) *What is the degree of usability of the PBL-Coach from the standpoint of the teachers and students?*". The Technology Acceptance Model (TAM) was used as an assessment tool.

II. MAIN THEORETICAL REFERENCES

A. PBL Principles

In [7], Savery & Duffy set out a set of eight instructional principles for PBL, namely: (i) Anchor all learning on activities to a larger task or problem; (ii) Provide support for engaging the student on the task or problem; (iii) Design an authentic task; (iv) Design the task and the learning environment to reflect the complexity of the environment for which the students should have the skills to interact in, at the end of the learning; (v) Give the student ownership of the process used to work out the solution; (vi) Design the learning environment to support students' thinking while challenging them; (vii) Encourage the testing of ideas against alternative views and contexts; (viii) Give opportunity for reflecting on learning and what has been learned. In addition, the authors in [ref] include more two PBL principles, completing a list of "10 PBL principles": (ix) PBL lays down a process of multi-directional teaching and learning; (x) PBL is supported by planning processes and continuous monitoring. Regarding the CMCPBL features commented in Section 1, the principles (i), (ii), (iii), (iv) and (vi) are related to feature (1), that emphasize the importance of students deal with a real and relevant problems into an authentic learning environment; while the principles (v), (vii) and (viii) are related to feature (2), with focus on collaborative knowledge; and (ix) and (x) related to feature (3), that reinforce the interactions among all participants of the learning environment.

B. xPBL

Although PBL is based on principles, it does not define a single methodology for their application within an educational setting. In [3], the authors put forward a methodology called *xPBL*, with the aim of ensuring that when PBL is employed for the teaching of Computer Studies, it is implemented in an effective, authentic and rigorous manner. According to the authors, the authenticity of the learning environment for PBL can be preserved if the following factors are taken into account: 1) the adoption and practice of real problems; 2) the definition of the human resources involved, together with their clearly defined functions and responsibilities; 3) theoretical content aligned with problem- solving; and 4) the adoption of the kind of development processes and assessment procedures employed by the market; 5) those involved are within an environment that reflects the reality of the job market. By preserving the learning environment and faithfully adhering to the guidelines of the PBL methodology, what is taught to the students can be rendered much more effective as a means of preparing them for their professional lives.

In [1] and [8], the authors show that it is possible to ensure that PBL is adopted in an effective way when it follows the stages of a well-defined process that encompasses planning, execution, monitoring and assessment and is geared towards making continuous improvements. These stages rely on the PDCA cycle (Plan, Do, Check and Act), which is a methodology that is basically designed to assist in the diagnosis, analysis and prognosis of organizational problems, and is ideally suited to the management of the teaching/learning process in PBL.

C. Delisle solving-process

In order to help students to better understand the problem chosen and propose a more adequate solution to it, was developed a dynamic that made use of Delisle problem-solving model [9]. The model is composed of four aspects that must be observed: 1) Ideas: possible solutions to the problem; 2) Facts: information about the problem; 3) Hypotheses, identification of learning problems to solve the problem and; 4) Plan of Action: strategies, information resources and other information that lead to the resolution of the problem. Figure 1 illustrates the solving-process defined by Delisle.

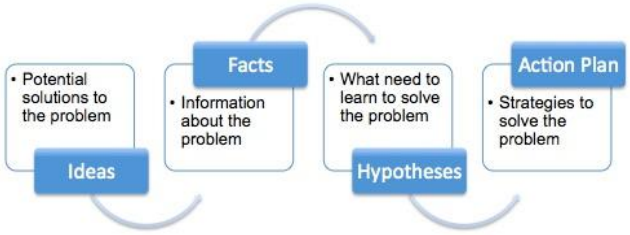


Figure 1. Solving-process by Delisle.

III. METHOD: DESIGN SCIENCE RESEARCH

The guidelines for the scientific method which shapes the different stages of this research study, can be found in Design Science Research (DSR), a research method which involves analyzing the use and performance of artifacts that are designed to understand, explain and improve the behavior of specific factors in the domain of Information Systems [10]. The basic principle of Design Research is that the knowledge, understanding and solving of a problem are acquired in the construction and application of an artifact within the context of a particular problem. Therefore, the DSR method was adopted in five steps:

1) *Understanding the Problem*: to obtain a clearer understanding of the teaching method. The aim of this activity was to find out about the principles and factors that, in the view of several authors, govern the PBL method. The second activity was to highlight the challenges and any particular obstacles that might face the PBL method. As a result, it was possible to draw up a list of problems regarding the management of PBL, such as the following: the difficulty of setting out a procedure to assist the students with problem-solving, the complexity of assessment, and how to teach managerial skills to students with a technical background, among other factors.

2) *Suggestions step* was to make conjectures about how technology can be used to facilitate learning in the PBL approach. The second activity was to design a model for problem-solving; this had in mind the xPBL methodology commented in Section II.B, which establishes specific means of panning and implementing PBL in the teaching of Computer Studies. As a result, a conceptual model of the learning environment was originated, as shown in Section IV.

3) *Development step* was to define the architecture, tools and development processes. This led to the design of the software artifacts of the PBL-Coach assessed by real users, as described in Section V.

4) *Assessment step* was to understand the preparation, application and analysis of the artifacts, together with the end users, with the aim of determining, in the first moment, the usability and utility of the PBL-Coach. This resulted in the setting out of performance measurement standards.

5) Conclusion stage was to understand what we learn. It should be mentioned that the assessment procedure foresees future iterations.

IV. PBL-COACH IN PDCA CYCLE

The conception of PBL-Coach was based on the main references commented in Section II: the ten PBL principles; the xPBL methodology; and the Delisle solving-process. As in PBL, the use of PBL-Coach follows a process composed of prescriptive activities. These activities allow the management of the teaching and learning process, since they rely on the PDCA cycle as foundation. Moreover, there is a relation between these activities and the PBL principles, explicitly pointed out in next sections. Figure 2 illustrates the process of using PBL-Coach within the planning, execution monitoring and improvements steps, highlighting the activities of each one.

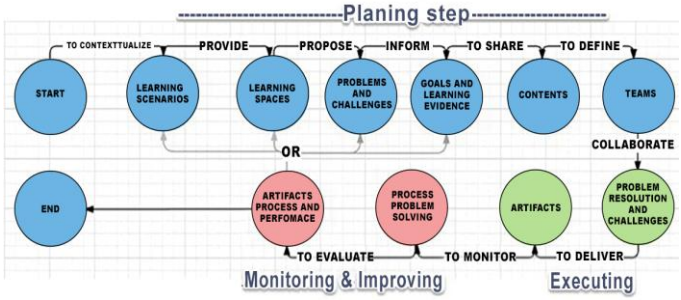


Figure 2. Interface to manage problem and challenges by teachers.

In the Planning step, six activities stand out: "*learning scenarios*", in which the teacher/tutors defines a learning scenario that correspond to real circumstances and to allow that students can reflecting on a situation where they should be enabled to assimilate interests of their learning; "*learning space*", where the teacher/tutors should make a place available for the students where they can do research and share technology, especially designed to encourage the collaboration among them (3D interface, as shown in Figure 4); "*problems & challenges*", when the teacher set real problems that challenge the students to seek to explore matters, and challenges that represent a partial goal in the solving-process controlled by schedules and deliverables; "*goals and evidences of learning*", when the teacher defines the learning objectives, describing observable knowledge and the skills required from the students; "*content*", related to sharing of text-based files, presentations, spreadsheets and so on; and "*teams*", when teacher/tutor form heterogeneous teams in which the different skills of each member can be complemented to achieve results.

Regarding Execution Step, two activities were defined: "*solving of problem & challenges*", Where students discuss and debate their knowledge about potential solutions, extract facts about the problem, define hypotheses about the challenges (according to Delisle's problem solving process) and also define the tasks that are required to solve the problem (action plan), dealing with the different phases of the workflow; and "*artifacts*", that consist in creation of web-based documents, such as spreadsheets and presentations, among other kinds of collaborative work.

Finally, the Monitoring and Improving steps define two activities: "*monitoring resolutions*", when the teacher/tutors assess artifacts, procedures and performance produced by students; and "*continuous improvements*", when the teacher should find out what difficulties students have, deal with any queries, provide feedbacks and check the deliverables of the teams.

V. USING PBL-COACH IN SOFTWARE ENGINEERING EDUCATION

This section describes the use of the PBL-Coach in the Agile Project Management module of an undergraduate course in Software Engineering. There is a total of 60 contact hours over 4 months (March-June 2016), the educational goal being to train 16 students (organized in four teams) in good practices in Agile Project Management.

To conduct the course, planning was carried out by the pedagogical team comprising three teachers with project management and PBL experience. This plan followed the guideline proposed by xPBL (mentioned in Section II).

A. Planning

In the Planning step, the teachers had access to a specific URL of PBL-Coach. The real life client (the owner of the problem) can "submit problems" by giving the name and e-mail address and by publishing a video or file in a PDF format, so that it is contextualized. In this case, each team was supported by one real client, who presented a list of problems related to software solutions. The teachers created learning scenarios and "chose the most effective problems". After this initial selection, the teachers were able to bring together their educational objectives by giving guidance to the students with regard to their learning goals and their choice of strategies to achieve them, as well as the information that would provide support for their chosen problem-solving.

Figure 3 shows the interface for the management of problems and challenges. This resource makes it possible to store problems in a shared repository and carry out searches that cover a wide range of information of interest. It will also make it possible to visualize the problems through specific information (presentations and video provided by real clients, for example), as well as to plan challenges to solve problems

(into a specific period) or choose new problems on the basis of pre-existing problems.

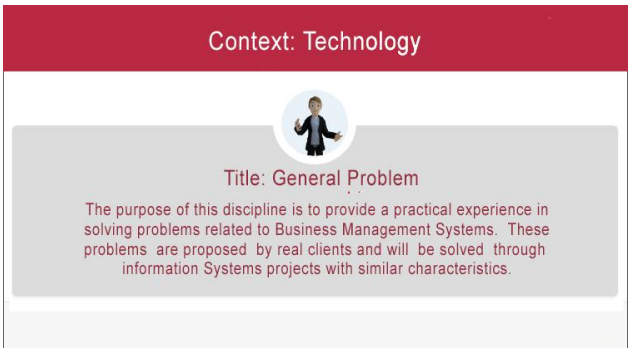


Figure 3. Interface to manage problem and challenges by teachers.

Following this, the teachers created immersive learning environments (learning spaces) and invited people or ¨human capital¨(students, teachers and clients – depending on the role they played) to form learning groups. This resource is based on the model for learning environments of Google Learning Space (Figure 4), promoting meeting and sharing of technological resources of the PBL-Coach.

Figure 4. Interface to manage problem and challenges by teachers.

All this "human capital¨that had been invited, was notified by e-mail and supplied with information about access to the virtual learning scenario in PBL-Coach.

B. *Execution*

In the execution step (Do), the students had access to PBL-Coach and began to see the problems that had been set. Moreover, they were able to clear up any uncertainties about the terms, situations, words or expressions they were unable to understand. They also visualized and coped with the challenges and learnt to understand what subjects or issues needed to be studied to overcome them. This understanding entailed identifying learning objectives/issues. Following this, the students revealed their knowledge and investigated facts, hypotheses and ideas with regard to the challenge-problem, through an interface called "analysis of solutions" based on Delisle process described in Section II.C. Figure 4 shows the interface for the analysis of solutions for problems-challenges. This resource provides a board, which establishes the analytical procedure for monitoring how students find solutions to problems. It also offers the students a means of thinking about a problem in depth and reaching a conclusion by means of the following sequence of solving-process proposed by Delisle.

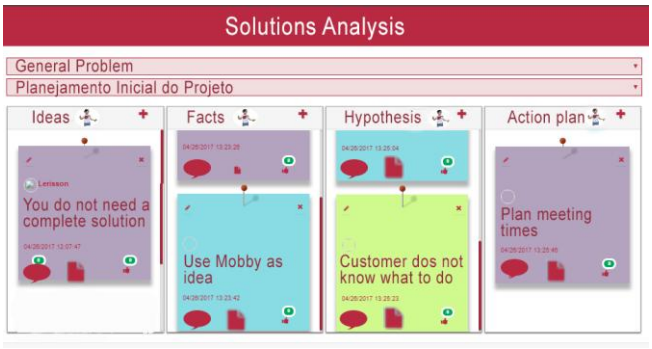


Figure 5. Analysis of solutions board.

Each member of the group could remember different things and this turned the discussion into an opportunity to learn, posting opinions directly in the analysis solutions board. It was important to show respect for the opinions of other people. The next step was to define the tasks, which could meet the expectations of the team, elaborating an action plan in a task management board. Figure 6 shows the interface for task management. This resource adopts the agile Kanban approach, which shows the workflow stages with the activity attributes of members of the team. In PBL-Coach, the Kanban board is organized in columns for the following stages: "to do", "doing", "done", "checked" and "obstacles". In this way, the boards are divided into columns, which represent the current state of each task. Each team must to manage its board, defining tasks and executors for them. Its value is that it can still operate as a management approach for agile practices and promoting self-management skills.

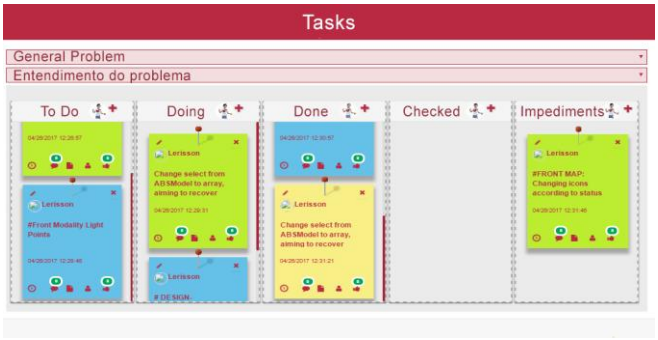


Figure 6. Interface to manage problem and challenges by teachers.

These tasks were transparent in so far as all the necessary information was disclosed to all human capital involved in teaching and learning process (students, teachers, clients), improving the communication and integration among them. The students also create documents, spreadsheets and presentations in a collaborative way, as well as holding remote meetings with the members of their teams, teachers and clients. About meetings, the students can also access an immersive learning environment (learning space), as shown in Figure 4.

C. Monitoring and Improving

In the monitoring and improvement stages (Check and Act), the teachers identified any difficulties the students might have in solving the challenge-problems, analyzing results or carrying out tasks. With the teachers constant support and supervision, along with the "like" button and comment sections in the cards at PBL-Coach, the ideas that the students posted in "Analysis of solution board" there could be continuously validated. This way, we were able to give positive feedback, guiding the students towards the right path, from the moment they found their first ideas to the moment they actually finished their projects.

They could also assess the degree of involvement of each group or each student with regard to the activities and conduct a 360 degree evaluation. A 360 degree evaluation took place between all students. This kind of evaluation was chosen because it provides a self-evaluation, plus a pairs and teachers evaluation. We also opened up for verbal feedback and suggestions from the students on the project as a whole. PBL-Coach also automatically generates a graph that quantifies how much the students actually entered data, commented and got positive feedback in the software. This production of visual graphs is actually one of the outputs for the process element guidelines proposed in [11].

VI. Analyzing Utility and Usability of PBL-Coach

The assessment stage of the research was characterized as descriptive. It involved data collection, which was carried out by means of a questionnaire based on TAM (Technology Acceptance Model) that was answered by sixteen students and three teachers who made use of the PBL-Coach, described in Section V. The questionnaire with 8 questions was available by Google Forms.

A. Utility

The purpose of utility evaluation is to assess the compliance of PBL-Coach with the PBL approach, regarding its principles. Figure 7 brings together all the results obtained in the questionnaire with regard to the utility dimensions. A multi-choice system was adopted where the questions are set out as alternatives variables and the respondents must choose one of them. This was based on the Likert Scale, which has a format that distinguishes between the following categories: 1) I Strongly Agree, 2) I Agree, 3) I Don´t Know, 4) I Disagree and 5) I Strongly Disagree.

The first and second question asked the interviewees about: Q1) *if PBL-Coach provides support for the research question, analysis and problem-solving, and addresses the challenges*; and Q2) *if it makes immersive learning spaces available for the students so that they can be involved in learning practice, as well as activities where they are given feedback by their peers and teachers*. It can be observed that for Q1 66.67% of the answers were obtained the "I strongly agree" category, and 33.33% in "I agree", whereas on the case of Q2, 50.00% were obtained within the "I strongly agree" category and 50.00% within "I agree". This level of agreement was maintained in both variables, which means that PBL-Coach helped to extend knowledge by giving support to problem-solving. The system also provides a virtual learning environment which allowed the students to work closely together; this because this learning strategy keeps them in contact with their peers when addressing differences of opinion arising from the answers of the interviewees.

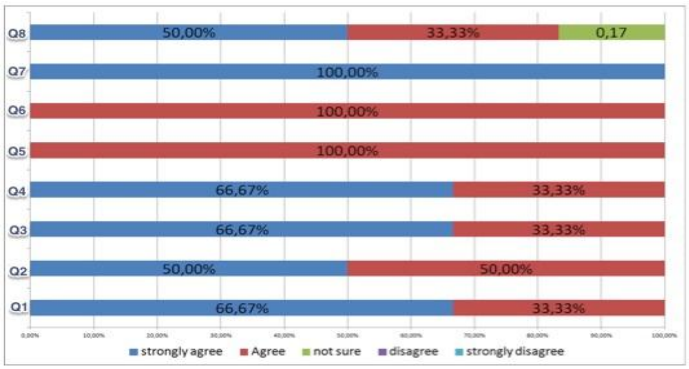


Figure 7. Utility evaluations of PBL-Coach.

The third and fourth variables asked the participants: Q3) *if the PBL-Coach allows to include (and have access to) information*; and Q4) *if it allowed an exchange of information among all involved in teaching and learning process, so that they could help each other in a way that fostered a multi-directional approach to teaching*. It is worth noting that in both Q3 and Q4, the answers obtained related to the "I strongly agree" (66.67%) and "I agree" (33.3%) categories. This level of agreement was maintained in both the affirmations, which means that PBL-Coach can assist in ensuring collaboration and the sharing of information.

The fifth, sixth and seventh questions asked the interviewees respectively: Q5) *if PBL-Coach enables the students and teachers to communicate with each other*; Q6) *if it assisted the students and teachers in planning their activities*; and Q7) *if it gave the students the freedom to make decisions about the problem that was set and how to reach a solution*. It should be noted that in Q5 and Q6, the answers obtained were related to the "I Agree" category (100%), and "I strongly agree" (100%) for the question Q7. The level of agreement was maintained in all the variables, which means that PBL-Coach can lead to greater interaction between the teacher and student in different situations that occur both inside and outside of the classroom, using the PBL-Coach. This can encourage students to adopt a critical spirit of inquiry and stimulate active learning.

Finally, the eighth question asked the participants if the PBL-Coach benefits the monitoring and improvement of the way learning is constructed. It can be seen that, most of the answers obtained, related to "I Strongly Agree" (50.0%) category, "I Agree" (33.3%) and ¨I Don´t know¨ (16.67%). The level of agreement was maintained in both affirmations, which means that PBL-Coach benefits the monitoring of the way students undertake their learning. The reason for this is that, the PBL-Coach follows a series of steps proposed by xPBL methodology that must be undertaken, or in other words, it involves procedures and graphical resources, that makes it easier to assess the performance of the students.

B. Usability

The usability evaluation concerns to assess the degree to which the users believed that the PBL-Coach is ¨usable¨ during

the APM course. This evaluation was carried out in three cycles, with clear evolutions in each one. Many aspects were considered in this point, such as the perception of use, interface use, standardize of terms, system performance, ease of use, and so on. Here, only general perceptions are showed, due space limitations. Improvements were implemented in the system at the end of each cycle, based on these assessments. Figure 8 shows the evaluation related to general reactions to the PBL-Coach. As is shown, the average growth related to general reactions was 2.7 in the first cycle, growing to 3.8 in the second cycle and 3.9 in the third cycle, considering the scale shown in the graphic of Figure 8 (terrible – excellent; frustrating-satisfactory; tedious-stimulant, and so on). This result showed a positive perception of participants from the APM course, commented in Section V.

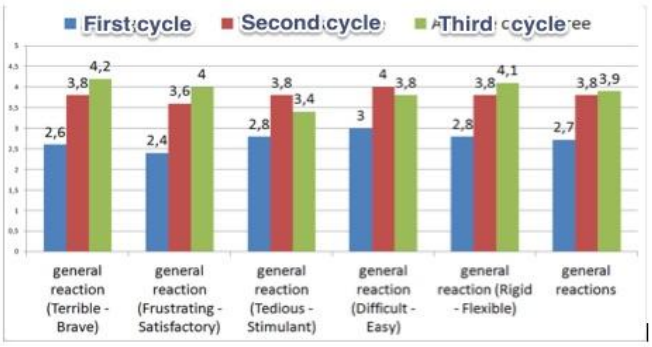


Figure 8. General reaction about usability.

Figure 9 shows the evaluations related to use of PBL-Coach: "start to use", "learning curve to use" and "general perception of use". The graph of Figure 9 indicates relevant evolutions in each of these dimensions, very close to the positive extreme of the scale (for example, difficult-easy).This result indicated a good level of acceptance and indicates the easily to learn to use the PBL-Coach.

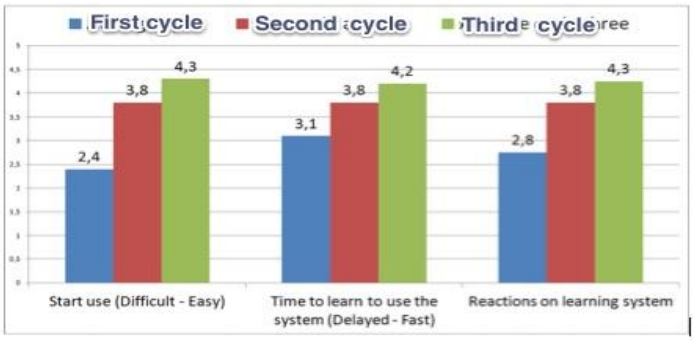


Figure 9. Interface to manage problem and challenges by teachers.

VII. CONCLUSIONS

Considering the effectiveness of the PBL method, this paper presented a virtual learning environment with interactive, technological resources, called PBL-Coach, the aim of which is to oversee the implementation of the PBL method and ensure that it adheres to its principles. A case study yielded results that were favorable to the PBL-Coach, which it validated together with a group of users. In general terms, it was clear that PBL-Coach proved to be suitable for attaining its objectives. On the basis of the results obtained from perceived utility, it can be concluded that the use of PBL-Coach makes it possible to oversee the implementation of the PBL approach. This approach is also able to cater for the needs of a learning environment that is centered on the students and aligned with a learning methodology, management processes and collaborative tools. It should also be underlined that PBL-Coach does not only give support to the planning, execution and monitoring of learning but also foster solutions analysis and task management skills.

Finally, in spite of the positive results for the initial assessment of usability and utility, it should be stressed that, although the environment is a necessary and powerful condition, it is not the only feature required for an effective learning process. It is the activities carried out by students during the teaching/learning process that determine to what extent learning has taken place and the involvement of the teacher in the environment also helps to foster this learning. In view of this, there are new kinds of ongoing validation that are being undertaken from the perspective of the utility and usability of this environment in real-life cases.

PBL-Coach is currently being used in three educational programs at a professional and academic level, and this confirms its importance since it has created situations of interest for teachers who have decided to use this environment in the schools and universities where they teach.

REFERENCES

[1] Figuêredo C.O., Santos S.C.,Alexandre,G.H.S.andBorba P. H. M.(2011), "Using PBL to develop Software Test Engineering", CATE, Cambridge, UK.

[2] Oliveira, A. M.; Santos, S. C.; Garcia, V. C. . PBL in Teaching Computing: An overview of the Last 15 Years. FIE, Oklahoma, EUA., 2013.

[3] Santos S. C., Furtado F., Lins W. "xPBL: a Methodology for Managing PBL when Teaching Computing", FIE, Madrid, Spain, 2014.

[4] Kahiigi, E.K., ekenberg, L., hansson, H., tusubira, F.F., danielson, M. 2008. Exploring the e-Learning State of Art. The Electronic Journal of e-Learning 6. 77 – 88.

[5] Savin-baden, M. 2003. Facilitating Problem-Based Learning. McGraw-Hill Education (UK). 170 pages.

[6] Scardamalia, m.; Bereiter, c. Adaptation and understanding: A case for new cultures of schooling. International perspectives on the design of technology-supported learning environments, Mahwah, NJ: Erlbaum., p. 149 – 163, 1996.

[7] Savery, J.R.; DUFFY, T. M. Problem Based Learning: An instructional model and its constructivist framework. Educational Technology, 1995, 35, p. 31-38.

[8] S. C. Santos and A. Pinto, "Assessing PBL with Software Factory and Agile Processes", CATE, Naples, Italy, 2012.

[9] Delisle, R (1997). How to use problem-based learning in the classroom. ASCD: Alexandria, Virginia, EUA.

[10] Vaishnavi, V.; kuechler, W. Design research in information systems. 2004.Britain, S. and Liber, O. 1999. A Framework for Pedagogical Evaluation of Virtual Learning Environments. Environments.

[11] S. C. dos Santos. "PBL-SEE: An Authentic Assessment Model for PBL-Based Software Engineering Education". Transaction on Education. IEEE, 2016.

Knowledge-based Learning Content Generation in the StudyBattles Environment

Amal Shehadeh, Alexander Felfernig, Martin Stettinger
Institute for Software Technology
Graz University of Technology
Graz, Austria
{ashehade,alexander.felfernig,martin.stettinger}@ist.tugraz.at

Michael Jeran, Stefan Reiterer
SelectionArts GmbH
http://www.selectionarts.com
Graz, Austria
{m.jeran,s.reiterer}@selectionarts.com

Abstract– *E-learning environments provide an orthogonal approach to transfer relevant knowledge. For example, sales representatives can improve their sales knowledge more independently from related courses offered. Major challenges for successfully establishing e-learning technologies in a company are to develop learning content in an efficient fashion, to recommend only relevant content to system users, and to motivate them to utilize the learning environment in a sustainable fashion. In this paper, we present the gamification-based e-learning environment* StudyBattles. *We provide an overview of* StudyBattles *functionalities including content creation, gamification techniques, learning performance analysis, and automated question generation. We show how* StudyBattles *is and can be utilized for different learning purposes in academic and professional environments. In addition, we introduce an approach to automatically generate product and sales domain learning content from recommender knowledge bases to be exploited in* StudyBattles. *Finally, we report the results of an initial qualitative study related to the applicability of* StudyBattles *in different domains, the potential improvements that* StudyBattles *can achieve, and additional functionalities that should be integrated.*

Keywords– *gamification-based e-learning; automated question generation; knowledge-based recommender systems; constraint satisfaction problem; knowledge acquisition*

1 INTRODUCTION

E-learning [13] provides an orthogonal approach to knowledge transfer in sectors such as companies, public organizations, universities, and schools. Companies apply e-learning environments for establishing a *corporate memory* that can be (and often has to be) used by employees to improve their process-relevant knowledge (e.g., marketing, sales, and product-specific knowledge). Similarly, in the public sector, employees establish process-relevant knowledge often on the basis of e-learning environments. Although it's also applicable to transfer process-relevant knowledge, employees at universities and schools apply e-learning environments primarily for the purpose of improving the quality of teaching and to reduce related time efforts.

Improvements triggered by the application of e-learning environments are manifold. E-learning environments make learning material widely accessible to users and thus create *flexibility with regard to the time of learning and training*. For organizations, it becomes *easier to understand strengths and weaknesses of employees with regard to organization-related knowledge*.

An entry barrier to the usage of e-learning systems, especially in companies, is the absence of relevant content, i.e., additional investments are needed before being able to ramp up the system. An approach to tackle this challenge is to provide *crowd sourcing* concepts which allow expert users on their own to provide the relevant content. This approach avoids knowledge acquisition bottlenecks since knowledge acquisition tasks become decentralized.

Content creation can be made even more efficient by *automatically generating learning content from different types of knowledge sources* [11]. In the remainder of this section we will shortly analyze existing approaches to content generation and then focus on the content generation approach implemented in StudyBattles.

Agarwal et al. [2] introduce an automatic question generation system that can generate gap-fill questions from a document. Gap-fill questions are fill-in-the-blank questions with multiple choices provided (one correct answer and three distractors). The system selects the most informative sentences of the chapters and generates gap-fill questions thereof by first blanking keys from the sentences and then determining the distractors for these keys. Syntactic and lexical features are used in this process without relying on any external resource apart from the information in the document.

Hussein et al. [8] propose a system which generates questions from a document by selecting one sentence at a time, extracts sections of the source sentence, then applies transformation rules for constructing a question. The proposed system is based on the OpenNLP open source statistical parser to generate the questions.

Gütl et al. [7] introduce a concept and prototype implementation of the Enhanced Automatic Question Creator (EAQC) which extracts most important concepts out of textual learning content and creates single choice, multiple-choice, completion exercises, and open ended questions on the basis of these concepts. Their approach combines statistical, structural, and semantic methods of natural language processing as well as a rule-based solution for concept extraction. EAQC is designed to deal with multilingual learning material.

Afzal [1] introduces an approach to automatically generate multiple-choice questions from text. They extract semantic relations and then automatically generate questions using these semantic relations. They evaluated the whole system, for example, in terms of question understandability, usefulness of semantic relations, and overall multiple choice question usability and found out that the generated questions are considered as relevant.

Alsubait et al. [3, 14] present an approach to automatically generate multiple-choice questions from OWL ontologies. They present an empirical evaluation of an ontology-based question generation approach and examine the feasibility of applying ontology-based question generation by educators with no prior experience in ontology building. They found out that this approach can result in a reasonable number of educationally useful questions.

In this paper, we introduce an approach that has commonalities with the question generation approach introduced in [3, 14]. In the StudyBattles environment, a *recommender knowledge base* can be used for question generation where questions do not only refer to the *solution space* defined by the knowledge base but also to situations where *no solution can be found* for a given set of customer requirements. Compared to the approach presented in [3, 14], our question generation approach is based on a recommender knowledge base represented as a Constraint Satisfaction Problem (CSP) [4] which is exploited for automated question generation.

Our contributions in this paper are the following. First, we provide an overview of the StudyBattles e-learning environment. Second, we introduce a set of new techniques that can be applied for *automated question generation out of constraint-based recommendation knowledge bases*. Finally, we report initial results of a qualitative

DOI reference number: 10.18293/SEKE2017-093

study related to the applicability and effectiveness of StudyBattles.

The remainder of this paper is organized as follows. In Section 2, we provide an overview of the major functionalities of the StudyBattles e-learning environment. In Section 3, we present a working example from the domain of automotive services (we selected this domain as a working example because it is easy to understand). In Section 4, we introduce an approach to the automated question generation from recommender knowledge bases – these questions can then be exploited by StudyBattles. An empirical evaluation of the functionalities provided by StudyBattles is presented in Section 5. The paper is concluded with a discussion of issues for future work in Section 6.

2 StudyBattles Environment

StudyBattles is an e-learning environment that has been developed within the PeopleViews research project[1] at Graz University of Technology. The StudyBattles start screen is depicted in Fig. 1 which includes a short system description, a list of subscriptions to learning applications, and further learning applications to subscribe to. StudyBattles mobile clients are available as iOS, Android, and HTML-5 web client. The system itself is available as global platform[2] and as an in-house server solution for individual enterprises. StudyBattles learners can join communities and subscribe to learning applications where they can practice exercises, add content, and compete against each other in quiz-based duels. StudyBattles supports content categorization in the learning applications, allowing for better structuring and easy information access for learners.

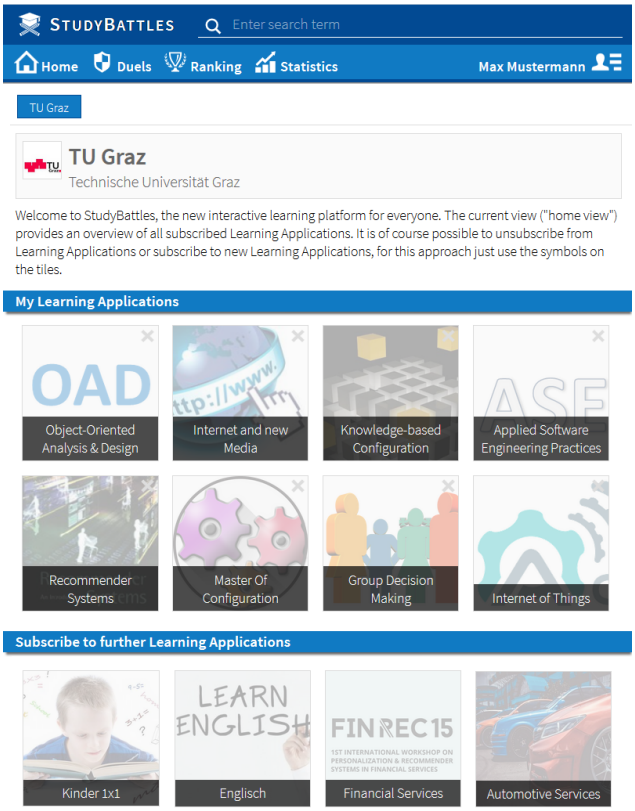


Figure 1. StudyBattles start screen (HTML-5 version) consisting of a set of learning applications that can be selected by the user.

StudyBattles *deployments*. The system is already deployed and applied at one municipality and two universities in Austria. At the mentioned universities, StudyBattles is applied in three Software Engineering courses and in two Artificial Intelligence related courses. The goal of the StudyBattles instance deployed at the mentioned Austrian municipality is to *increase employees knowledge with regard to security-related topics* and to transfer application-oriented knowledge related to *a new accounting system*. Currently, StudyBattles is also deployed for one of the largest financial service providers in Austria. The major goal of this deployment is to support sales representatives in their learning processes related to *product knowledge and sales practices*. StudyBattles users from these domains have been interviewed within the scope of our qualitative study reported in Section 5. Furthermore, StudyBattles is currently used within the scope of StudyBattles:Help project[3] to help refugees to quickly learn important and needed topics related to their integration into the Austrian society.

Learning and training. After subscribing to a StudyBattles learning application, the user can select categories and questions to answer. After answering a question, the user receives an immediate feedback on the correctness of his or her answer. If the answer is incorrect, a related explanation is provided to the user (in case it has been included during the definition phase of the question).

Content Creation and Question types. StudyBattles follows the concept of *crowd sourcing* where users can enter questions and expert users can evaluate the quality of the questions. The status of a *domain expert* is reached if a certain threshold of correctly answered questions is passed. Users are allowed also to add additional content in terms of text, documents, pictures, and movies which serve as a basis for answering questions.

StudyBattles supports different types of questions, examples are depicted in the Figures 2– 5. Fig. 2 depicts a simple example of a *multiple-choice question* – the question is related to the relationship between customer requirements and corresponding products (cars). The abbreviations used in Fig. 2 represent customer requirements: pu = *purpose of use*, cat = *category*, and fc = *fuel consumption*.

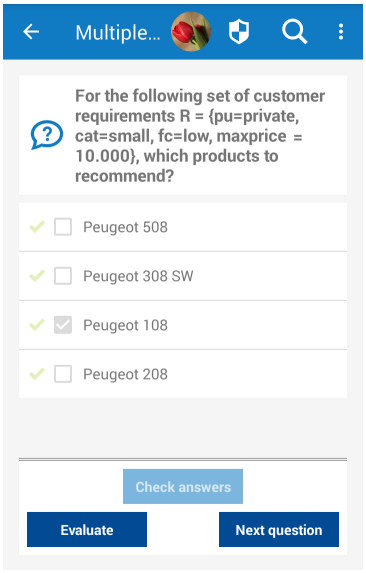


Figure 2. StudyBattles: representation of multiple-choice questions (Android version). The green check marks represent the correctness feedback (pu = *purpose of use*, cat = *category*, and fc = *fuel consumption*).

Fig. 3 depicts an example of an *association task* where terms on the right-hand side have to be associated with the corresponding terms on the left-hand side. The corresponding HTML-based definition interface is depicted in Figure 6. Association tasks can be exploited, for example, for **(1)** asking questions regarding compati-

[1] The work presented in this paper has been conducted within the scope of the PeopleViews project funded by the Austrian Research Promotion Agency (843492).

[2] www.studybattles.com

[3] helpers.studybattles.com

bility relationships between customer requirements and products. **(2)** asking questions about the incompatibility of specific customer requirements. In the example of Fig. 4 users are requested to associate inconsistent customer requirements on the left and right hand side.

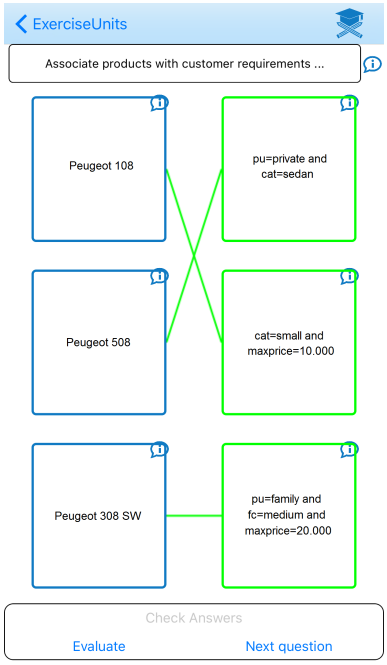


Figure 3. STUDYBATTLES: representation of an associations task (iOS version) – products on the left have to be associated with corresponding compatible requirements on the right.

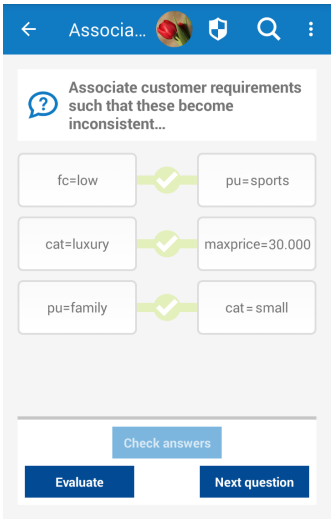


Figure 4. STUDYBATTLES: Association task related to the association of inconsistent requirements (Android version).

In Fig. 5 a question is posed to educate users with regard to repair knowledge (which situations lead to an empty set of solutions and how can a user recover from such situations).

Gamification. STUDYBATTLES supports several gamification techniques such as points, levels, ranking, badges, feedback, and duels. Users can trigger quiz-based duels and then are assigned randomly to opponents. Alternatively, users can play against the best performing user in a certain learning application – the ranking of a user is visible "locally", i.e., a user is only able to see other users directly ranked before and after him/her. STUDYBATTLES points can be collected if a user is the winner of a duel. There are other opportunities to collect STUDYBATTLES points, for example, when answering a question correctly or adding a new question to the system.

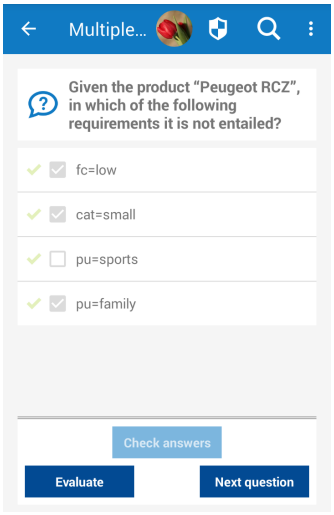


Figure 5. STUDYBATTLES: representation of inconsistency-related knowledge in terms of a multiple-choice question.

The higher the estimated complexity of the question, the higher is the number of received STUDYBATTLES points. The complexity of a question can be evaluated directly after having answered the question (see the Evaluate button, for example, in Fig. 5)

Analysis of learning performance. STUDYBATTLES supports different types of statistics (for users and administrators) that help to analyze the strengths and weaknesses of the user community and to establish needed counter measures (e.g., improving/adapting some parts of the learning material). The statistics are provided on the Learning application, categories, and user levels.

3 A SIMPLE AUTOMOTIVE SERVICE ADVISOR

Knowledge-based recommender systems [9] are conversational systems where users are enabled to specify their preferences and the system proposes a corresponding solution. Constraint-based recommender systems [4] are a specific type of knowledge-based recommenders where recommendation knowledge is defined in terms of a constraint satisfaction problem (CSP) – see the following definition.

Definition (Recommendation Task). A recommendation task can be defined as a CSP (V, D, C, REQ, $PROD$) where V is a set of variables describing potential customer requirements and product properties, D represents domain definitions for the variables, C is a set of constraints, REQ represents a set of customer requirements, and $PROD$ is the products catalog.

Definition (Recommendation). A recommendation (solution) for a given recommendation task ($V, D, C, REQ, PROD$) is a complete set A of variable assignments $v_i = a$ to the variables $v_i \in V$ ($v_i = a \rightarrow a \in \text{domain}(v_i)$) with *consistent*($A \cup C \cup REQ \cup PROD$).

Consequently, solutions determined for recommendation tasks can be considered as candidate recommendations for a customer (user). Alternative solutions can be further ranked according to their utility for the customer – for details on how to rank such candidate sets we refer to [9]. We now introduce a simple *Automotive Service Advisor knowledge base* that will be used as working example throughout the paper. The variables in V are *purpose of use* (pu), *category* (cat), *fuel consumption* (fc), *maxprice*, *price*, and $name$ which represents the name of the car.

- $V = \{pu, cat, fc, maxprice, price, name\}$
- D = {domain(pu) = {family, private, sports}, domain(cat) = {sedan, coupe, luxury, MPV[4], small}, domain(fc) =

[4] Multi-Purpose Vehicle (MPV): a car classification used in Europe to describe small family cars.

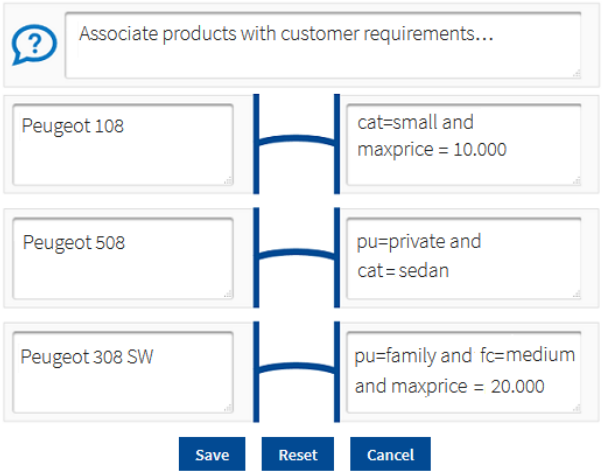


Figure 6. STUDYBATTLES: definition of an association task (specification of correct pairs) in the HTML-5 version.

{low, medium, high}, domain($maxprice$) = {10.000, 20.000, 30.000, 40.000, 50.000}, domain($price$) = [9.000-50.000], domain($name$) = {Peugeot 5008, Peugeot 308 SW, Peugeot 108, Peugeot 208, Peugeot 508, Peugeot RCZ, Peugeot 308 GTi, Peugeot 4008 380THP}}

- $C = \{c_1 : price <= maxprice, c_2 : cat = small \rightarrow fc = low, c_3 : pu = family \rightarrow cat \notin \{small, coupe\}, c_4 : cat = luxury \rightarrow fc = high, c_5 : pu = private \vee pu = sports \rightarrow cat \neq MPV, c_6 : cat = small \rightarrow price < 12.000, c_7 : \neg(pu = sports \wedge fc = low), c_8 : cat = luxury \vee pu = sports \rightarrow maxprice \in \{40.000, 50.000\}$
- $REQ = \{r_1 : pu = private, r_2 : fc = low, r_3 : maxprice = 10.000\}$
- $PROD$ is the product catalog which represents the constraints responsible for restricting the possible instantiations of the variables in V. A simple example of a product catalog for our automotive recommender is shown in Table 1.

Table 1. product catalog (pu = *purpose of use*, cat = *category*, and fc = *fuel consumption*).

name	pu	cat	fc	price
Peugeot 5008	family	MPV	medium	27.030
Peugeot 308 SW	family	MPV	medium	18.213
Peugeot 108	private	small	low	9.399
Peugeot 208	private	small	low	11.971
Peugeot 508	private	sedan	high	28.503
Peugeot RCZ	sports	coupe	high	49.990
Peugeot 308 GTi	sports	sedan	high	36.143
Peugeot 4008 380THP	family	luxury	high	37.543

A recommendation for the given example recommendation task is the set of variable assignments $A = \{pu = private, cat = small, fc = low, price = 9.399, name = Peugeot108\}$. In this situation, the given set of customer requirements defined in REQ are consistent with the constraints in C. However, even a slight change in the specification of REQ can lead to a situation where the customer requirements become inconsistent with the constraints in C. For example, if we define $REQ = \{r_1 : pu = family, r_2 : fc = low, r_3 : maxprice = 10.000\}$, no solution can be found.

In situations where no solution can be found for a given set of requirements, concepts of model-based diagnosis [12] can help to identify a minimal set of requirements that have to be adapted such that a solution can be identified. For the identification of such minimal changes, we introduce the definition of a diagnosis task and a corresponding diagnosis (see the following definitions).

Definition (Diagnosis Task). A diagnosis task is defined as a tuple $(C, REQ, PROD)$ where C is a set of constraints, REQ is a set of customer requirements, $PROD$ is the product catalog, and $REQ \cup C \cup PROD$ is inconsistent.

Definition (Diagnosis). A set $\Delta \subseteq REQ$ for a given diagnosis task $(C, REQ, PROD)$ is a diagnosis if $REQ - \Delta \cup C \cup PROD$ is consistent, i.e., Δ is a set of requirements (constraints) that have to be deleted from REQ such that the remaining customer requirements are consistent with $C \cup PROD$. Furthermore, Δ is minimal if there does not exist a set Δ' with $\Delta' \subset \Delta$.

The basic algorithm for determining minimal diagnoses is introduced in [12]. For further algorithms specifically designed to be applied in interactive settings we refer to [5]. All these algorithms are complete, i.e., they are able to determine the complete set of minimal diagnoses on the basis of the concepts of hitting set directed acyclic graphs introduced in [12] – this is an important property than can also be exploited in the context of question generation (see the following section).

Types of sales knowledge. Knowledge-based recommender systems include sales knowledge in different forms. First, given a set of customer requirements, a recommender can determine items that can be recommended (*filter knowledge*). Second, given an item, a recommender can determine customer requirements that are consistent with the item (*product knowledge*). Third, in situations where no solution can be identified for a given set of customer requirements, diagnosis can determine the needed minimal changes to help the user out of the *no solution could be found dilemma* (*analysis knowledge*). Fourth, given an item, a recommender can determine a set of customer requirements which are inconsistent with the item (*inconsistency knowledge*). In the following, we discuss how recommendation task definitions can be exploited for the generation of questions for an e-learning environment that is used, for example, for the education of sales representatives.

4 GENERATING QUESTIONS FROM RECOMMENDATION TASK DEFINITIONS

Questions and corresponding answers for the STUDYBATTLES environment can be automatically generated from the constraints contained in a recommendation task definition. On the basis of our working example, we now introduce an approach to automatically generate questions and related answers for the four aforementioned types of sales knowledge. The generated questions and answers can be imported into STUDYBATTLES and then be used for training and supporting sales representatives. The overall goal of these questions is to increase the personal level of sales knowledge and, as a consequence, to make advisory services more efficient.

4.1 *Filter knowledge related question generation*

The underlying task is to figure out which items fit a given set of predefined customer requirements (REQ). More formally, filter knowl-

edge related questions can be generated on the basis of a recommendation task definition (V, D, C, REQ, $PROD$). On the basis of such definition, a constraint solver (recommender) is able to determine all possible instantiations of customer requirements and corresponding items. Each set of customer requirements can then be the basis of the question, the corresponding items represent the correct answer(s).

Example. Given the recommendation task definition of our working example (Automotive Services Advisor), a set of customer requirements could be $REQ = \{r_1 : cat = MPV, r_2 : fc = medium, r_3 : maxprice = 20.000\}$, the set of corresponding correct answers is $\{name = Peugeot308SW\}$. The corresponding question that would be posed in the StudyBattles environment is: ***Given the following customer requirements ..., which item(s) would you recommend?***

4.2 Product knowledge related question generation

The underlying task is to figure out which sets of customer requirements fit a given item. In this context, for each possible item a constraint solver can return the complete collection of sets of customer requirements which are consistent with this item.

Example. Given the recommendation task definition of our working example, the item selected could be $name = Peugeot308SW$. The corresponding collection of consistent customer requirements is $\{R_1 : \{r_1 : pu = family, r_2 : cat = MPV, r_3 : fc = medium, r_4 : maxprice = 20.000\}, R_2 : \{r_1 : pu = family, r_2 : fc = medium, r_3 : maxprice = 20.000\}, R_3 : \{r_1 : pu = family, r_2 : cat = MPV, r_3 : maxprice = 20.000\}, R_4 : \{r_1 : cat = MPV, r_2 : fc = medium, r_3 : maxprice = 20.000\}$, i.e., in this example there exist *4* sets R_i of customer requirements that are consistent with the selected item $Peugeot308SW$. The corresponding question that would be posed in the StudyBattles environment is: ***Given the following item ..., which sets of customer requirements are consistent with that item?***

4.3 Analysis knowledge related question generation

The underlying task is to figure out which minimal sets of customer requirements have to be adapted such that a solution can be found. More formally, analysis knowledge related questions can be generated on the basis of a diagnosis task definition ($C, REQ, PROD$). The diagnosis task definition can be used for question generation, the related answers are represented by the determined diagnoses Δ_i.

Example. Given the recommendation task definition of our working example with an adapted set of customer requirements $REQ = \{r_1 : pu = sports, r_2 : cat = coupe, r_3 : fc = low\}$, which is inconsistent with ($C \cup PROD$), the corresponding alternative minimal sets of customer requirements (diagnoses Δ_i) that have to be deleted from REQ such that a solution can be identified, are the following: $\{\Delta_1 = \{r_1, r_2\}, \Delta_2 = \{r_3\}\}$. For example, deleting (or adapting) the requirement r_3 restores consistency, i.e., allows the calculation of a recommendation. The corresponding question that would be posed in StudyBattles is: ***Given the following recommendation task definition ... which one is a minimal set of requirements that have to be deleted from*** REQ ***such that a recommendation can be identified?***

4.4 Inconsistency knowledge related question generation

The underlying task is to figure out which sets of customer requirements trigger an inconsistency with a pre-selected item. More formally, inconsistency knowledge related questions can be generated on the basis of a recommendation task definition (V, D, C, REQ, $PROD$) where all combinations of customer requirements have to be determined that never include the pre-selected item.

Example. In our working example, we could pre-select the item $name = PeugeotRCZ$. Combinations of customer requirements that do not entail the item $name = PeugeotRCZ$ are all possible combinations with the exception of the following combination: $\{pu = sports, cat = coupe, fc = high, maxprice = 50.000\}$. The corresponding question that would be posed in StudyBattles is: ***Given the following item ... which combination of customer requirements does not entail this item?***

5 StudyBattles Evaluation

In order to evaluate the applicability and effectiveness of the StudyBattles environment, we conducted a qualitative study with experts N=20 from different domains (financial services, public administration, telecommunications, and universities). In this study, domain experts provided feedback related to major potential improvements that come along with the application of StudyBattles, the possible domains of application, and what additional functionalities should be integrated into the system[5].

Potential improvements due to the application of StudyBattles. Within the scope of this study, the potential improvements that have been pointed out are the following: improved knowledge retention in organizations (See Fig. 7), improved knowledge sharing between users on the basis of community-based (crowd-sourced) knowledge acquisition processes, increased motivation to learn, improved skills, increased fun and interest in the topic, increased competition level between users (e.g., sales representatives), improved quality of service with regard to customers (e.g., due to the already discussed question generation mechanisms - see Fig. 8, reduced time efforts in the learning process and services (e.g., sales representatives and advisory services in the automotive domain - see Fig. 9), and enhanced possibilities of community knowledge analysis which provide a basis for a more fine-grained adaptation of learning material where needed.

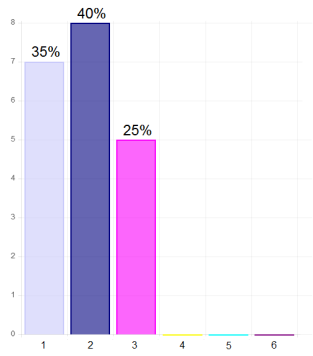


Figure 7. Statistics: Effectiveness of StudyBattles - Improved knowledge retention in the organization: (1) Excellent, (2) Very Good, (3) Good, (4) Neutral, (5) Fair, (6) Poor

Application scenarios. In addition to the application domains where StudyBattles has already been deployed, the participants of the study mentioned the following additional application scenarios where StudyBattles could be applied: in the *health domain*, doctors have additional means to keep their knowledge up-to-date and also to get confronted with new health-related knowledge in a more systematic fashion. *University personnel* can apply StudyBattles not only for teaching purposes but also for providing e-learning content to new employees (e.g., PhD students in their first semester who are in need of learning about the internal processes and rules). *Manufacturers* (e.g., in the automotive domain) can apply the system to improve the production processes and sales related knowledge of employees. Further mentioned application domains are *schools*, *societies* (intuitive explanation of rules for new members), and a *global*

[5] Statistics that illustrate the results cannot be displayed all in the paper for space reasons.

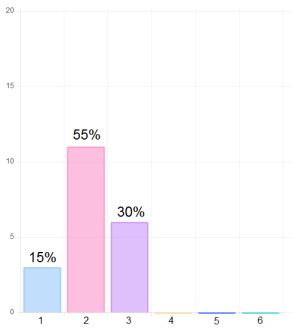


Figure 8. STATISTICS: Effectiveness of StudyBattles - Improved quality of services.

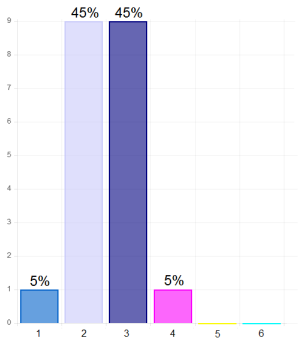


Figure 9. STATISTICS: Effectiveness of StudyBattles - Reduced time efforts in services.

platform for knowledge provision and consumption that can be applied by everyone.

Additional functionalities. Another focus of the qualitative study was to figure out additional functionalities that should be integrated into STUDYBATTLES. Functionalities mentioned by the participants are the following: with the generated questions (see Section 4), a more fine-grained determination of STUDYBATTLES points could be achieved, for example, by also taking into account the prediction quality of sales representatives (in one scenario, questions are related to the filter knowledge of sales representatives – the better the predictions are, the more corresponding points should be assigned to the user). *Social network* functionalities should be included since these provide an additional means to achieve a more frequent usage of STUDYBATTLES, for example, due to personalized status messages. Beside the already included personalized ranking of questions, additional recommendation functionalities should be included, for example, the recommendation of opponents in duels, the recommendation of potential friends in the mentioned social network, and the recommendation of additional learning content. Finally, participants of the study proposed the inclusion of intelligent user interface elements that help to increase the engagement in STUDYBATTLES, for example, a kind of traffic light feedback interface could motivate more frequent interaction with the system (see also [10]).

6 CONCLUSIONS

In this paper, we provided an overview of the e-learning environment STUDYBATTLES and how it can be used as a complementary approach to transfer knowledge related to product and sales practices.

Artificial Intelligence concepts currently included in the STUDYBATTLES environment are basic recommendation mechanisms related to the selection of questions to be shown to users, and mechanisms that support the automated generation of questions and related correct answers. Future work will focus on an extension of the provided recommendation functionalities, for example, by recommending users for duels, recommending learning applications, recommending content in a collaborative fashion, and recommending users for an inclusion in specific learning teams. Furthermore, we will extend our approach to learning content generation, for example, by allowing to control the degree of complexity of the content. In this context, we will also analyze potential synergies with existing approaches to test case generation in software engineering.

Especially in the context of educating sales representatives, automated question generation becomes a key functionality, since it reduces the overheads of manual content generation and management, which is often the task of only a small group of persons. In future versions of STUDYBATTLES, additional question types will be included. For example, we will provide mechanisms that allow to generate not only questions related to diagnoses (analysis knowledge) but also to related repair actions (i.e., changes in the requirements that lead to the identification of at least one solution). In addition, we will use approaches to reduce the number of calculated diagnoses, such as *minimal cardinality diagnoses* [12][6], to identify the diagnoses that are more relevant to users (especially in the case of large sets of diagnosis alternatives). Furthermore, diagnosis personalization techniques [6] will be exploited to improve the diagnosis prediction quality, by integrating recommendation approaches such as similarity-based, utility-based, probability-based, and ensemble-based with standard model-based diagnosis [12].

REFERENCES

[1] N. Afzal, 'Automatic generation of multiple choice questions using surface-based semantic relations', *International Journal of Computational Linguistics (IJCL)*, **6**, 26–44, (2015).

[2] M. Agarwal and M. Prashanth, 'Automatic gap-fill question generation from text books', in *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, IUNLPBEA '11, pp. 56–64. Association for Computational Linguistics, (2011).

[3] Tahani Alsubait, Bijan Parsia, and Uli Sattler, 'Generating multiple choice questions from ontologies: lessons learnt', in *Proceedings of the 11th International Workshop on OWL: Experiences and Directions (OWLED 2014)*, pp. 73–84. CEUR-WS.org, (2014).

[4] A. Felfernig and R. Burke, 'Constraint-based recommender systems: technologies and research issues', in *Proceedings of the 10th International Conference on Electronic Commerce*, ICEC '08, pp. 3:1–3:10. ACM, (2008).

[5] A. Felfernig, M. Schubert, and C. Zehentner, 'An efficient diagnosis algorithm for inconsistent constraint sets', *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AIEDAM)*, **26**(1), 53–62, (2012).

[6] Alexander Felfernig, Monika Schubert, and Stefan Reiterer, 'Personalized diagnosis for over-constrained problems', in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pp. 1990–1996. AAAI Press, (2013).

[7] C. Gütl, K. Lankmayr, J. Weinhofer, and M. Höfler, 'Enhanced automatic question creator - EAQC: concept, development and evaluation of an automatic test item creation tool to foster modern e-education', *Electronic Journal of e-Learning*, **9**, (2011).

[8] H. Hussein, M. Elmogy, and S. Guirguis, 'Automatic English question generation system based on template driven scheme', *IJCSI International Journal of Computer Science*, **11**(1), (2014).

[9] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender systems: an introduction*, Cambridge University Press, 1st edn., 2010.

[10] G. Ninaus, A. Felfernig, M. Stettinger, S. Reiterer, G. Leitner, L. Weninger, and W. Schanil, 'INTELLIREQ: intelligent techniques for software requirements engineering', in *ECAI 2014 - 21st European Conference on Artificial Intelligence*, pp. 1161–1166. IOS Press, (2014).

[11] S. Rakangor and Y. Ghodasara, 'Literature review of automatic question generation systems', *International Journal of Scientific and Research Publications*, **5**, (2015).

[12] R. Reiter, 'A theory of diagnosis from first principles', *Artificial Intelligence Journal*, **32**(1), 57–95, (April 1987).

[13] L. Shen and R. Shen, *Advances in web-based learning – ICWL 2004: Third International Conference Proceedings*, chapter Learning content recommendation service based-on simple sequencing specification, 363–370, Springer Berlin Heidelberg, 2004.

[14] A. Tahani, B. Parsia, and U. Sattler, *Knowledge engineering and knowledge management: EKAW 2014. Revised Selected Papers.*, chapter Generating multiple choice questions from ontologies: how far can we go?, 66–79, Springer International Publishing, 2015.

Development of a Data Mining Education Framework for Data Visualization in Distance Learning Environments

Angelo F. Dias Gonçalves and Alexandre M. A. Maciel
University of Pernambuco - UPE
Recife, Brazil
{afdg, amam}@ecomp.poli.br

Rodrigo L. Rodrigues
Rural Federal University of Pernambuco
Recife, Brazil
rodrigo.linsrodrigues@ufrpe.br

***Abstract*— With the increasing interest in developing Learning Analytics tools that can be integrated into the well-known Moodle course management systems nowadays, many tools have already been developed. These tools usually requires the user to know data mining techniques, and also requires time to get mining results from the tools. To address this problem, in this article, we present a structure that uses pre-built data mining through Shiny to quickly obtain results with a focus on visualizing data with graphs, and thus allows the integration of other research, called FMDEV. Guided by the proposed framework, a tool was developed for display the data mining results in a few clicks for Moodle users who wish to have them for day-to-day use and allows users with more experience in data mining to integrate new research. Finally, we used FMDEV tool to generate some experimental results using a set of real-life sample data from undergraduate students.**

Moodle; Distance Learning; Framework; Educational Data Mining; tools.

I. INTRODUCTION

The last Brazilian census for learning distance in higher education was published in 2016. It was realized with 339 institutions coming from all administrative categories, federal, state and municipal, public, private and non-profit, institutions of the National Learning Service (e.g., Sesi, Sebrae, Senac, Senai etc.), non-governmental organizations (NGOs), third sector and public agencies [1].

According to EaD.BR 2015 Census [1], most of the professionals working at Distance Education (DE) are tutors and teachers, in which 29,380 tutors and 18,769 teachers. In EaD.BR 2014 Census [2], 17,692 tutors and 11,074 teachers were raised. It represents a growth of almost 60%. These professionals are responsible for teaching courses that currently have 5,048,912 students in Brasil [1].

A Learning Management System (LMS) gathers vast amounts of information about students' interactions. Most of this information refers to communication through existing tools such as discussion forums, chats, and digital media postings. These tools, when used, generate information on how and when students execute their assignments, tasks, course commitments, etc. The Moodle's default reporting tool offers information and filtering capabilities, but the data they provide is considered raw data and does not provide meaningful information about the teaching-learning process [3].

According to Luna, Castro, and Romero [4], extracting this type of useful data and transforming such information into actionable knowledge is a difficult task, and Educational Data Mining (EDM) offers both different and convergent perspectives, methodologies, techniques, and tools aiming to facilitate the process of discovering knowledge from this environments. EDM is defined as the development of methods for exploring the unique types of data that come from educational settings and using those methods to understand students and what and how they learn [5].

Nevertheless, currently, although existing tools offer different levels of analysis, basic graphs on user interaction with the platform, and some data mining techniques, manipulation of these tools still requires prior knowledge of data mining, and is time-consuming to use day to day. Based on this, it is necessary to expand this information extracted from data mining to teachers and tutors from all areas.

To overcome this drawback, we developed a particular framework, called visual educational data mining framework (in Portuguese: FMDEV), which enables the extraction of knowledge and exports it visually to a larger number of LMS users. Finally, to extend the framework objective, we developed a tool for the purpose of validating the framework, because tools like this are more precise and easier to be used by non-expert users in data mining than general or traditional data mining tools [7]. In this way, instructors can discover, in a simple way, hidden information about students' behavior, and how students learn in the courses in a single and user-friendly interface [8].

The objective of this study is to present a valid framework that can help the main users of LMS Moodle in daily decision making, independent of data mining knowledge, thus providing a tool and its experimental results. Therefore, this article is

DOI reference number: 10.18293/SEKE2017-130

organized as follows: Theoretical Foundation in Section II, Framework presentation in Section III, Validation of the Framework in Section IV, and finally, Conclusion and Future Works in Section V.

II. THEORETICAL FOUNDATION

A. *Data Mining and Educational Data Mining*

Data Mining (DM) can be considered as a central stage within the more general knowledge discovery process [6]. The objective of data mining is to determine models that include the relations among data and allow a better understanding, prediction or generalization of these data.

According to Fayyad, Shapiro and Smyth [9], the main idea of data mining techniques is feature selection and pattern recognition of database system. Data mining can be declared as the knowledge discovery from an enormous amount of raw data using statistical methods, machine learning, and artificial algorithms.

According to Romero and Ventura [5], Educational Data Mining is the field that discovers new knowledge based on students', teachers', tutors' and manager' usage of the system to improve the quality of the education offered through online courses. In their review of state of the art about EDM, they list eleven educational tasks that make use of data mining techniques.

Among these tasks we highlight two that are related to the objective of this work:

- 'Analysis and Data Visualization,' which aims to highlight useful information and support decision making.
- 'Providing Feedback for Supporting Instructors', which supports course authors/teachers/administrators and enables them decision making (about how to improve students' learning, organize instructional resources more efficiently, etc.) and allow them to take appropriate proactive and/or remedial action. It is important to point out that this task is different than data analyzing and visualizing functions, which only provide necessary information directly from data (reports, statistics, etc.).

B. *Shiny and Sharing Apps*

Shiny [10] allows the non-expert computer scientist to create publication-ready figures and tables through an intuitive interface to the underlying computer code.

The Shiny can build a web-based application which is published on the Internet. It is easy to develop and to integrate into a web content using HTML and CSS [11]. The Shiny is an R package which is responsive to display the results of the data mining analysis into web-based applications. This web-based application was developed using the R programming language which is open source with several attractive packages [12].

The Shiny has two structures, contained in the files server.R and ui.R. The server file is a set of instructions that build the R components while the user-interface file is a set of instructions to display the application. It is already possible to find in the literature several web-based applications using Shiny and with several techniques of data mining and algorithms [13] [14].

Sharing is the central part of Shiny applications. To publish your Shiny app, only organize the files and post to a server in the cloud. RStudio insists on this sharing by providing a free server, called shinyapps.io [15]. Shinyapps.io is a server maintained by RStudio, where it is possible to make online applications available for easy access and use. Through this server, it can access any shared application from anywhere.

III. FRAMEWORK

FMDEV tool has been developed in HTML5, CSS, and JavaScript language and integrated into Moodle as an HTML block. The main reason for integrating the proposed tool as an HTML block is because any of these HTML blocks provide the ability to create external links, which is desired by our tool.

Blocks are items which may be added to the left or right column, smaller sizes, or central, larger size, on any Moodle page. They contain information or new applications geared towards students or instructors. In our case, the added HTML block allows only the tutors/teachers to visualize the results exported by the framework and to have access to the administration of the framework through a tool button. Anyone responsible for administering LMS Moodle, before adding the block, must insert the framework package to the same server. In this way, it is easy to install FMDEV into existing Moodle deployments.

From a high-level viewpoint, FMDEV comprises four main modules (see Fig. 1).

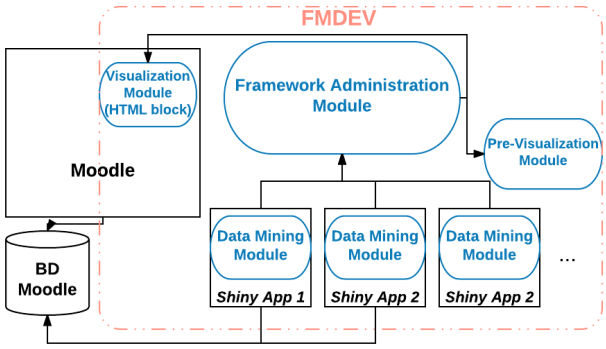


Figure 1. *FMDEV Architecture.*

FMDEV allows several data mining modules to be incorporated, where each module consists of a Shiny application, where they will be items in the administration list of the framework. Therefore, the proposed tool consists of three consecutive steps:

- The Pre-Visualization module enables the user to visualize the information that will also be shown in the visualization module and allows the user to return to the framework administration module, for a new selection, through the back button. It also allows the user to close the screen so that the user can return to Moodle.
- The visualization module, in the HTML block of Moodle, enables the display of the result of the first

module of data mining embedded in FMDEV. In this module, there is a button, in tool format, that when clicked the user is redirected to framework administration module.

- The Framework Administration Module enables the user to select one of the available data mining module options from the drop-down menu. When the user choice an option, he is redirected, in this step, to the pre-visualization module.

IV. VALIDATION

To validate FMDEV it was realized a benchmark with some found tools, based on the web, as well as an experiment with real data.

A. *Benchmark*

Currently, almost all new EDM tools are web based. We compare them with some examples next. Table 1 illustrates a comparison of the FMDEV with some web-based EDM tools.

This benchmark evaluated relevant points for our research, such as: Previous knowledge in data mining to visualize the results (Data M.); Allows inclusion of data mining projects beyond those already included in the package (Personal.); is it free? (Free); is it integrated to the LMS? (Integr.); and is it open source? (Open S.).

TABLE I. BENCHMARK OF EDM WEB-BASED TOOLS.

Tool	Data M.	Personal.	Free	Integr.	Open S.
FMDEV	No	Yes	Yes	Yes	Yes
GISMO [16]	Yes	No	Yes	Yes	Yes
SNAPP [17]	Yes	No	Yes	Yes	No
AAT [18]	Yes	No	No	No	No
MOClog [19]	Yes	No	Yes	Yes	No
E-learningWebMiner [20]	Yes	No	No	No	No
CVLA [21]	Yes	No	No	Yes	No
IntelliBoard.net [22]	Yes	No	No	Yes	No
SmartKlass [23]	Yes	No	Yes	Yes	No
MEAP [24]	Yes	No	Yes	Yes	Yes
Analytics graphs [25]	Yes	No	Yes	Yes	No
VeLA [26]	Yes	No	No	Yes	No

B. *Experiment*

For the purpose of validating the operation of the proposed FMDEV, a real use experiment was realized. The LMS used was the Moodle of the Nucleus of Distance Learning (NEAD) of the University of Pernambuco (UPE), so the data used was gathered from on People Management discipline, part of 3rd-year reading Pedagogy course, with almost 200 enrolled students.

In the tests done two mining modules, the "shiny" apps, were developed, in which one of them was created based on the work of Machado [27], which used clusters analysis. The teacher / tutor will only need to select one of the modules as previously mentioned, but the administrator must be able to generate / integrate new data mining modules.

Fig. 2 shows FMDEV block already included into Moodle (on the right side), where it corresponds to the extracted view of the selected mining module, and the administration button (Tool button), which is in the lower right corner.

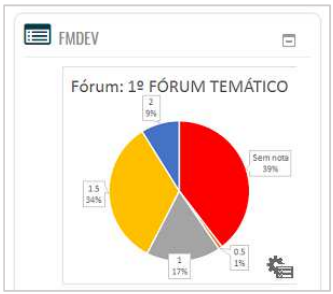


Figure 2. FMDEV block in a real course.

The developed Moodle block, as previously stated, has been developed in HTML5, CSS, and JavaScript language. It's possible to access the Framework Administration Module, also implemented as an HTML5 page. The administration page of the framework could be hosted on a server different from the one found in Moodle, just as the mining modules, shiny applications, were hosted on a separate server.

A simple front-end has been implemented, so the user does not need more than two clicks to select the desired option. In this implementation, the Shiny app are presented as options, allowing the choice between one of them in a drop-down menu in the administration page of the framework.

Soon after selecting the desired option, the user is redirected to the pre-visualization module, in which the same one that will be shown on this screen will be updated in the Moodle block. Fig. 3 shows the pre-visualization screen as described above.

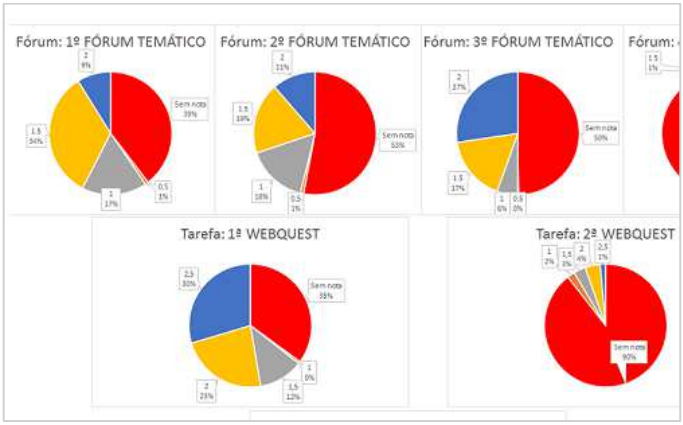


Figure 3. FMDEV pre-visualization module.

Finally, what is shown on this page will also be used to update the Moodle HTML block, previously shown in this work in Fig. 2.

V. CONCLUSIONS AND FUTURE WORKS

Moodle management systems include a growing number of data mining tools that provide various types of analysis and graphing of user interaction. However, it 's hard to find any integrated tool that allows you to obtain results quickly and without the cost of time, so that it can be used in the day to day routine of tutors and teachers. In this way, the objective of this

article is to propose a new architecture, implemented that architecture in the form of FMDEV, which it´s used as a tool allows the discovery of knowledge in Moodle data.

Finally, the utility of the proposed tool is described with an experiment using as consolidated LMS, and a real-life dataset of Pedagogy students. This experiment mainly illustrates the ease of obtaining data visually and quickly as the user chooses. The results obtained provide useful information so that users can provide faster feedback on how students' progress in distance learning courses.

In the future work, we intend to implement FMDEV in some real course and monitor the use of the tutors/teachers, giving support and adding new modules of data mining. We are currently working on the development of an improved visual interface, adapted to the size of the screen, to present the results obtained without the need of a scroll bar. Also, we would also like to implement many modules for mining academic papers, and, not least, to register our tool with the National Institute of Industrial Property (INPI in Brazil is the federal body charged with executing and enforcing laws regarding the regulation of Any Industrial Property).

ACKNOWLEDGMENT

The authors would like to thank the CNPQ for supporting the development of this work through the research projects granted by "Bolsa de Produtividade DT" (Process310752/2015-9).

REFERENCES

[1] ABED, Associação Brasileira de Educação A Distância, "Censo EaD.br: relatório analítico da aprendizagem a distância no Brasil 2015", 1 ed. Curitiba: InterSaberes, 2016.

[2] ABED, Associação Brasileira de Educação A Distância, "Censo EaD.br: relatório analítico da aprendizagem a distância no Brasil 2013", 1 ed. Curitiba: Ibpex, 2014.

[3] M. A. Conde, A. Hernádez-García, F. J. García-Peñalvo, and M. L. Séin-Echaluce, "Exploring student interactions: Learning analytics tools for student tracking", International Conference on Learning and Collaboration Technologies, 2015.

[4] J. M. Luna, C. Castro, C. and Romero, "MDM tool: A data mining framework integrated into Moodle", Computer Applications in Engineering Education, Vol. 25, 2017.

[5] C. Romero, and S. Ventura, "Educational Data Mining: A Review of the State-of-the-Art", IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews, Vol. 40, 2010.

[6] O. Maimon, L. and Rokach, "The Data Mining and Knowledge Discovery Handbook", 2nd ed., Springer, 2010.

[7] R. Mazza, M. Bettoni, M. Faré, L. and Mazzola, "MOCLog—Monitoring Online Courses with log data", 1st Moodle Research Conference Proceedings, Heraklion, Greece, 2012.

[8] S. Graf, C. Ives, N. Rahman, and A. Ferri, "AAT-A tool for accessing and analysing student's behaviour data in learning systems", In:Proceedings of the 1st International Conference on Learning Analytics and Knowledge, Lak, Banff, AB, Canada, 2011.

[9] U. Fayyad, G. P.-Shapiro, and P. Smyth, "From Data Mining to Knowledge Discovery in Databases", AI Magazine, 1996.

[10] Shiny by RStudio, "Easy web applications in R", Available from: https://www.rstudio.com/products/shiny/.

[11] C. Beeley, "Web application development with R using Shiny", Birmingham: Packt, 2013.

[12] W. Chang, J. Cheng, J. J. Allaire, Y. Xie, J. McPherson and M. Otto "Package 'shiny' ", 2015, Available from: http://shiny.rstudio.com.

[13] G. P. Siknun and I. S. Sitanggang , "Web-based classification application for forest fire data using the shiny framework and the C5.0 algorithm", 2016.

[14] R. Hermawati and I. S. Sitanggang, "Web-Based Clustering Application Using Shiny Framework and DBSCAN Algorithm for Hotspots Data in Peatland", Sumatra, 2016.

[15] Share your Shiny Applications Online, "Shinyapps.io by RStudio", Available from: http://www.shinyapps.io/.

[16] M. Luca, N. Mauro, M. Christian, and M. Riccardo, "L'aggiornamento del tool di monitoraggio delle attività degli studenti: GISMO 2.0", In: Atti del convegno italiano MoodleMoot, Bari, Italia, 2010.

[17] A. Bakharia, and S. Dawson, "SNAPP: A bird's-eye view of temporal participant interaction. International conference on learning analytics and knowledge", ACM New York, NY, USA, 2011.

[18] S. Graf, C. Ives, N. Rahman, and A. Ferri, "AAT-A tool for accessing and analysing student's behaviour data in learning systems", In: Proceedings of the 1st International Conference on Learning Analytics and Knowledge, Lak, Banff, AB, Canada, 2011.

[19] R. Mazza, M. Bettoni, M. Faré, and L. Mazzola, "MOCLog—Monitoring Online Courses with log data", 1st Moodle Research Conference Proceedings, Heraklion, Greece, 2012.

[20] D. García-Saiz, and M. E. Zorrilla, "A service oriented architecture to provide data mining services for non-expert data miners", Decis Supp Syst 55, 2013.

[21] B. Dragulescu, M. Bucos, and R. Vasiu, "CVLA: Integrating Multiple Analytics Techniques in a Custom Moodle Report", International Conference ICIST, Druskininkai, Lithuania, 2015.

[22] IntelliBoard, "IntelliBoard.net", 2015, Available from: http://www.intelliboard.net/.

[23] Klassdata, "SmartKlass: The Learning Analytics Plugin", 2015, Available from: http://klassdata.com/smartklass-learning-analytics-plugin/.

[24] D.Y.-T. Liu, J. Froissard, D. Richards, and A. Atif, "An enhanced learning analytics plugin for moodle: Student engagement and personalised intervention", In: Australasian society for computers in learning and tertiary education (ascilite 2015), Perth, Australia, 2015.

[25] J. Singh, "New Block: Analytics Graphs", 2015, Available from: http://www.Moodleworld.com/2015/03/31/new-block-analytics-graphs.

[26] J. Cruz-Benito, and F. Garcia, "VeLA: A visual eLearning analytics tool", LASI, Bilbao, 2015.

[27] C. J. R. Machado, B. R. B. Lima, A. M. A. Maciel, and R. L. Rodrigues, "An investigation of students behavior in discussion forums using Educational Data Mining", International Conference on Software Engineering and Knowledge Engineering (SEKE), Redwood City, San Francisco Bay, California, 2016.

Morpheme-Enhanced Spectral Word Embedding

Jiawei Liu
University of Science and Technology of China, China
ustcljw@mail.ustc.edu.cn

***Abstract*—Traditional word embedding models only learn word-level semantic information from corpus while neglect the valuable semantic information of words' internal structures such as morphemes. To address this problem, the goal of this paper is to exploit the morphological information to enhance the quality of word embeddings. Based on spectral method, we propose two word embedding models: Morpheme on Original view and Morpheme on Context view (MOMC) and Morpheme on Context view (MC). In vector space of MOMC and MC, both semantic-similar words and morphological-similar words locate near with each other. In experiments, MOMC, MC and the baselines are tested on word similarity and sentiment classification. The results show that our models outperform all comparative baselines on six datasets of word similarity and win the first on sentiment classification as well. Based on a large German corpus, we also inspect the ability of word embeddings to process morpheme-rich languages by using German word similarity task. The result shows that MOMC and MC significantly outperform the baselines more than 5 percentage on one dataset and nearly 4 percentage on the other. These impressive improvements demonstrate the effectiveness of our models in dealing with morpheme-rich languages like German.**

***Index Terms*—Spectral Method, Morphological Information, CCA, Random SVD, Word Embedding**

I. Introduction

Nowadays, Natural Language Processing (NLP) has been an important part of Artificial Intelligence (AI) for its effectiveness on many NLP tasks such as information retrieval [1] and text classification [2]. As we all know, computer can not directly deal with natural language due to the abstract semantic structure contained in corpus. To solve this problem, researchers developed a lot of models to represent words into vector space, which is also called word embedding.

Traditional word embedding models are divided into two main branches. One is based on neural network like Continuous Bag Of Words (CBOW) and Skip-gram [3]. The other is based on the matrix factorization such as the models in [4], which is also called spectral method. In general, the neural network-based models are famous for their robust performance. Nevertheless, these models with a large amount of parameters are time-consuming and tuning parameters requires enough experiences. Besides, CBOW and Skip-gram are also meaning-ambiguous methods, which lack theoretical explanation. On the contrary, the spectral models have their own advantages such as being on a theoretically provable basis and able to accelerate the calculation. Nevertheless, the models mentioned above are still word-level models and ignore many useful internal information of a word such as the morphemes. In English, a morpheme is the smallest unit and has some linguistic meanings. Mostly, morphemes can be divided into three categories. The prefix is one of the morphemes which has an affix placed before the stem of a word. Usually, adding a prefix before a word changes the meaning of the word but assigns similar semanteme. For example, putting *"uni-"* before the word *"form"* changes the sense of *"style"*, but all the words beginning with *"uni-"* will be endowed with the meaning of "single". The suffix is another morpheme which is an affix placed after the stem of a word. For instance, *"-ed"* is always added to the tail of the word to denote the past tense. Additionally, the root is the primary lexical unit of a word without prefix or suffix before or after it. In traditional models, the meaningful morphological information is abandoned.

Motivated by recent work, which exploits the internal structures to improve Chinese word embeddings [5], [6], in this paper, we utilize the spectral technique to incorporate the morphological information into word embedding training process. In our models, the word information, morphological information and context information are transformed into three matrices, respectively. Based on these matrices, we propose two models including Morpheme on Original view and Morpheme on Context view (MOMC) and Morpheme on Context view (MC) by using the Canonical Correlation Analysis (CCA). In MOMC, we combine the morphological information matrix with not only the word information matrix but the context information matrix. Then, CCA is conducted on these two new matrices. In MC, the morphological information matrix is only combined with the context information matrix. Then, we conduct CCA on the new combination matrix and the initial word information matrix.

In experiments, MOMC and MC together with the baselines are tested on word similarity and sentiment classification. The results demonstrate the advantages of our models, which outperform the baselines on six datasets of word similarity. Besides, MOMC and MC also achieve the best performance on sentiment classification. In order to evaluate the ability of our methods to deal with the morpheme-rich languages, we train all models on a German corpus and also test them on word similarity task. The result indicates that MC and MOMC significantly outperform the baselines on all the datasets and even get more than 5 percentage advantage on RG-65-German dataset. The performance of our models convinces us that incorporating morphological information into word embedding can generate a good structure in vector space and enhance the quality of word embeddings.

II. Related Work and background

As we mentioned above, there are a lot of word embedding models. In this section, we are going to give a brief review

DOI reference number: 10.18293/SEKE2017-105

of CBOW [3] and OSCCA (One Step CCA) [4], which are chosen as the baselines in experiments.

CBOW With a slide window, CBOW [3] utilizes the context words in the window to predict the target word. Given a sequence of tokens $T = \{t_1, t_2, \cdots, t_n\}$, the goal of CBOW is to maximize the following average log probability equation:

$$L = \frac{1}{n}\sum_{i=1}^{n} \log\ p(t_i|context(t_i)), \tag{1}$$

where $context(t_i)$ means the context information of t_i in the slide window. Based on *Hierarchical Softmax* and *Negative Sampling* [7], Equation (1) can be solved efficiently.

OSCCA Based on CCA, Dhillon et al. proposed a spectral word embedding model named OSCCA, which is proven to be effective in [4]. For a sequence of tokens $T = \{t_1, t_2, \cdots, t_n\}$, in OSCCA, we firstly need to construct two matrices including word information matrix W and context information matrix C. Building of these matrices will be introduced in the following section. Then, we directly do CCA on these matrices. As we all know, the goal of CCA is to find a pair of projection matrices Φ_w and Φ_c such that the correlation between the projection of W onto Φ_w and C onto Φ_c is maximized. Based on *Eigendecomposition*, the solution of Φ_w and Φ_c is as follows.

$$\begin{aligned} \Sigma_{11}^{-1}\Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\Phi_w &= \lambda\Phi_w \\ \Sigma_{22}^{-1}\Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}\Phi_c &= \lambda\Phi_c \end{aligned} \tag{2}$$

where Σ_{11} is the covariance matrix of W, Σ_{12} is the covariance matrix between W and C, Σ_{22} is the covariance matrix of C, Σ_{21} if the covariance matrix between C and W. Nevertheless, this solution is time-consuming and memory-consuming due to many large sparse matrix multiplications, which origins from the huge vocabulary size. To solve this problem, in [4], they demonstrated that the solution of OSCCA can be transformed into an equivalent equation.

$$\Sigma_{11}^{-1/2}\Sigma_{12}\Sigma_{22}^{-1/2} = \Phi_w\Lambda\Phi_c^T \tag{3}$$

where (Φ_w, Φ_c) are the left and right singular vectors and Λ is the diagonal matrix of singular values. Hence, $\Phi_w^{proj} = \Sigma_{11}^{-1/2}\Phi_w$ and $\Phi_c^{proj} = \Sigma_{22}^{-1/2}\Phi_c$. In OSCCA [4], the projection Φ_w^{proj} is viewed as the word embedding. For clarity, we define the anterior CCA progress as $(\Phi_w^{proj}, \Phi_c^{proj}) \equiv CCA(matrix_1, matrix_2)$ and use it if we want to do CCA on a pair of matrices. Due to the effectiveness of the second solution, all spectral word embeddings are trained based on Equation (3) in this paper.

III. Incorporating morphological information into word embedding

Based on spectral technique, we propose two word embedding models: Morpheme on Original view and Morpheme on Context view (MOMC) and Morpheme on Context view (MC). The objective of these models is to incorporate morphological information to improve the quality of word embeddings so that both semantic-similar words and morphological-similar words can group together in vector space. Both MOMC and MC are built based on three information matrices including word information matrix, context information matrix and morphological information matrix. For clarity, we introduce how to build these matrices firstly.

A. Methodology of Building Information Matrices

1) Notation: We define the word information matrix as $W \in \mathbb{R}^{n\times p}$, morphological information matrix as $M \in \mathbb{R}^{n\times(np+ns+nr)}$, left context information matrix as $L \in \mathbb{R}^{n\times cp}$, right context information matrix as $R \in \mathbb{R}^{n\times cp}$ and whole context information matrix as $C \in \mathbb{R}^{n\times 2cp}$ where n is the length of the word sequence, p is the number of the vocabulary, np is the number of the prefix, nr is the number of the root, ns is the number of the suffix and c is context window size.

2) Word Information Matrix: The word information is mapped into a sparse matrix W. Given a corpus $\{t_1, t_2, t_3, ..., t_n\}$, the vocabulary is $\{null, v_1, v_2, ..., v_p\}$ where $n \geq p$. Every word is represented as a one-hot vector in which $w_{ij} = 1$ means word t_i is in the j_{th} position of the vocabulary. In practice, the vocabulary is so huge that the computational cost is rather high. Since quite a number of words only turn up at a low frequency due to the *zipfian* distribution, we set a threshold of the frequency of the words. If the word's frequency is under that threshold, the word will be assigned to the first column *null*. More detail is give in Equation (4).

$$w_{ij} = \begin{cases} 1, \ when\ t_i = v_j\ or\ t_i = null \\ 0, \ otherwise \end{cases} \tag{4}$$

3) Morphological Information Matrix: In this paper, morphemes are divided into prefix, suffix and root. Therefore, the whole morphological information matrix M can be represented as the concatenation of three separate matrices. We assume that $root = \{null, r_1, r_2, \cdots, r_{nr}\}$, $prefix = \{null, p_1, p_2, \cdots, p_{np}\}$, and $suffix = \{null, s_1, s_2, \cdots, s_{ns}\}$. If a word has a certain morpheme, the corresponding column position should be set as 1. A constraint is that in the whole matrix, every morpheme should be active at least once. If there is no feature in the morpheme list, the first column *null* will be set as 1. We define function $match(t_i, a_j) \in \{1, 0\}, a_j \in \{\ prefix \cup root \cup suffix\}$ to simplify our description. The function means that if word t_i contains morpheme a_j, it will be set as 1 or 0 otherwise. Hence, we can utilize Equation (5) to summarize the construction of morphological information matrix.

$$m_{ij} = \begin{cases} 1, \ when\ match(t_i, a_j) = 1 \\ 0, \ otherwise \end{cases} \tag{5}$$

4) Context Information Matrix: Context information matrix C is composed of the left context matrix L and the right context matrix R. For a target word, the construction of the left context matrix is built based on the contextual words, which are located in the left-side of the target word. Building of the right context matrix is associated with the right-side contextual words of the target word. For a single token in corpus, the corresponding positions of its context words in the left context matrix and right context matrix will be set as 1. If the context word is not in the vocabulary, then the corresponding position of $null$ will be set as 1. We define a function $context(t_i, v_j, k),\ k \in [1, c],\ j \in [1, c \times p]$. When the k_{th} context word of t_i is equal to v_j, this function is equal to 1 or 0 otherwise. The mathematical description of

Algorithm 1 MOMC model

function MOMC (D, M, h, k)
Input: huge size corpus D
morpheme set M
context window size h
dimension of word embedding k
Output: word vector ϕ_w^{proj}
Initialization: Initialize expanded word information matrix W_e and context information matrix C_e according to the input parameters.
Calculate the approximate correlated matrix
$C_{ww} = W_e^T W_e$
$C_{wc} = W_e^T C_e$
$C_{cc} = C_e^T C_e$
Do SVD on
$C_{ww}^{-1/2} C_{wc} C_{cc}^{-1/2}$
Get *right* singular matrix ϕ_{w_e}
Get the eigenword $\phi_{w_e}^{proj} = C_{ww}^{-1/2} \phi_{w_e}$
Extract the vocabulary part skipping morpheme part ϕ_w^{proj}
Return ϕ_w^{proj}

left context matrix L is in Equation (6). Right context matrix can be established in the same way.

$$l_{ij} = \begin{cases} 1, \ when\ context(t_i, v_j, k) = 1 \\ 0, \ otherwise \end{cases} \tag{6}$$

B. Morpheme on Original view and Morpheme on Context view (MOMC)

In MOMC, the morpheme information is viewed as a unit feature. It means that all words including the target words and context words need to be featured by morpheme information. MOMC uses the expanded word matrix $W_e = [W\ M]$ and expanded context matrix $C_e = [[L\ M]\ [R\ M]]$ as the input. [ ] means concatenating two matrices upon row direction. Like what we did above, we utilize the following equation to describe the MOMC in a mathematic way.

$$(\Phi_w^{proj}, \Phi_c^{proj}) = CCA(W_e, C_e) \tag{7}$$

The detailed algorithm is described as the pseudocode in Algorithm 1 shown below. This model will lead words with similar morphemes to locate closer than that in the original vector space. Due to the fact that morphemes have some semantic roles, the method encoding the morphological information into the word vectors will be useful to identify words' semantic role and morphological role. Hence, it will have a better performance than *OSCCA* ideally.

C. Morpheme on Context view (MC)

This method is similar to *MOMC* which also exploit the internal structures of words. Nevertheless, this method deals with the morphological information in an entirely different way. In MC, the morphological information is viewed as a global feature rather than unit feature in MOMC, which acts as a unique character about the corpus. We don't represent words into the other space but take their context words and morphemes as the context. The solution of this model is also based on *CCA* model. The pseudocode of MC is similar to Algorithm 1. The only difference is the input matrix. For brevity, we don't give the algorithm of MC. The equation is like the MOMC which is shown as follows:

$$(\Phi_w^{proj}, \Phi_c^{proj}) = CCA(W_e, C_e) \tag{8}$$

where $W_e = W$ and $C_e = [L\ R\ M]$.

D. Random SVD for CCA

In [4], the authors discussed how to calculate Σ_{11}, Σ_{12} and Σ_{22} in Equation (3). Based on *zipfian* distribution, in OSCCA, the empirical expectations of all words are set as zero so that matrices $W^T W$, $W^T C$ and $C^T C$ can be utilized to stand for Σ_{11}, Σ_{12} and Σ_{22}, respectively. As we all know, a word and its morpheme are occurred simultaneously, which means the distribution of morpheme in large corpus follows the *zipfian* distribution as well. Hence, the empirical expectations of all morphemes are also set as zero so that we can utilize the same way to calculate the Σ_{11}, Σ_{12} and Σ_{22} in MOMC and MC.

As we mentioned above, getting word embeddings needs to train a huge corpus. Doing decomposition on so vast matrix still needs a great deal of time and memory even though the computing power has become more powerful. Hence, recent advances in SVD algorithms are necessary to be taken to solve the huge matrix decomposition. In [8], Halko et al. gave a powerful tool which uses random projections to do SVD on large matrices. Basic idea of the method is to find a lower dimensional basis for large matrix and then to calculate the singular vectors in this lower dimensional basis. For a large input $m \times n$ matrix A, the first stage is to find a basis Q, based on which A can be represent as following equation.

$$A \approx QQ^T A \tag{9}$$

where Q has few columns. When calculate Q, we firstly generate a Gaussian test matrix Ω with dimensions of $n \times (k+l)$ where k is a target number of singular vectors and l is the extra basis vector ranging from 0 to k. By multiplying alternately with A and A^T, we secondly generate a matrix $Y = (AA^T)^q A\Omega$ where q is another parameter from input. Lastly, matrix Q whose columns form an orthonormal basis for the range of Y will be constructed. The second stage of this method is to do SVD algorithm on A with the help of Q. In this stage, a matrix $B = Q^T A$ needs to be calculated firstly. Then, SVD algorithm is conducted on the small matrix $B = U' \Sigma V^T$. Finally, the left singular matrix U of A can be approximately set as $U = QU'$. More theoretical demonstrations are illustrated in [8].

E. Complexity Analysis

The solutions of our models including MOMC and MC are based on the random SVD algorithm which is introduced in the previous subsection. Random SVD algorithm as the most important part of our models consumes most of the training time. For a large matrix A with the dimensions $m \times n$, the first stage needs to cost $O(mns)$ time to generate a low dimension basis Q with the dimensions $m \times s,\ s \ll n$. In [4], it is reported that the computational complexity of OSCCA is $O(p^2 cs)$. For MC, the cost of time will be $O(p(pc + np + ns + nr)s)$. Because np, ns and nr stand for the counts of morphemes

which are the constant numbers and much more smaller than pc, $O(p(pc+np+ns+nr)s)$ is equal to $O(p^2cs)$. Obviously, the complexity of MOMC is also $O(p^2cs)$. The notations p and c represent the vocabulary size and context window size which have been introduced. Compared with OSCCA, it is obvious that our models have same computational complexities.

IV. Experiments

In this section, we test MOMC, MC and the baselines on word similarity and sentiment classification. Moreover, parameter analysis is given in the end of this section. For clarity, some experimental settings are introduced firstly.

A. Experimental Settings

In this paper, all word embeddings are trained based on the news corpus of 2009, which is listed on the 2013 ACL Workshop on Machine Translation[1] and also used in [9]. We collect the morpheme from the website[2] and get 90 prefixes, 241 roots and 64 suffixes.

The existing word embedding model OSCCA [4] and CBOW [3] are chosen to compare with MOMC and MC. For a fair and unbiased comparison, all word embeddings are trained in the same condition. The size of the context window and the dimension of word embedding are set as 2 and 200, which are the same as the settings in [4]. CBOW is trained by using the source code[3]. For efficiency, the negative sampling algorithm is chosen to solve CBOW [7]. To generate the word embedding of OSCCA, we utilize the java toolkit, which is released in Dhillon's github[4]. Moreover, we modify the source code of this toolkit and generate our embeddings.

B. Word Similarity

This experiment is utilized to evaluate the ability of word embeddings to capture semantic information from large corpus. The dataset is composed of two parts: word pairs and human score. We need to calculate the similarities of word pairs firstly and then measure the correlation between the similarities and human score. The similarities of word pairs are measured using cosine distance. We use *Spearman's rank correlation coefficient (ρ)* to evaluate the correlation between word similarities and human score. Apparently, bigger ρ means better performance. In this task, we utilize 8 widely used benchmarks. RG-65 [10] has 65 noun pairs. MTurk-287 and MTurk-771 [11] contain 287 and 771 English word pairs, respectively. RW-STANFORD [12] has a large number of rare words with similarity score. WS-353-ALL [13] contains 353 pairs of English with human similarity ratings. WS-353-REL and WS-353-SIM are annotated based on WS-353-ALL in [14]. MEN-TR-3k [15] owns 3000 word pairs, which frequently turn up in a large web corpus.

The results are shown in Table I. It is obvious that our methods significantly outperform the comparative baselines. MOMC wins the first on four dataset and MC performs the best on other two datasets. The advantages of MOMC and MC are corresponding to our expectation. Obviously, more semantic information means better performance. The baselines are word-level models and neglect the internal semantic information. On the contrary, MOMC and MC exploit the morphological information and capture more semantic information, which interprets the best performance of our methods.

TABLE I
Results of Word Similarity and Sentiment Classification. "SA" stands for Sentiment Classification. The numbers in bold mean the best answers.

	OSCCA	MOMC	MC	CBOW
MTurk-287	0.5664	**0.5847**	0.5657	0.5761
RG-65	0.5674	0.5734	0.5734	**0.6042**
RW-STANFORD	0.4966	0.4798	**0.5107**	0.4961
WS-353-ALL	0.5387	**0.5742**	0.5634	0.5675
WS-353-REL	0.4486	**0.4607**	0.4338	0.4420
WS-353-SIM	0.6752	0.6827	**0.7072**	0.6970
MEN-TR-3k	**0.6562**	0.6339	0.6562	0.6346
MTurk-771	0.5324	**0.5532**	0.5338	0.5478
SA	0.7086	0.7132	**0.7256**	0.7193

C. Sentiment Classification

This experiment is conducted in a similar way as we find in [16]. The average of the word embeddings of a given sentence is utilized as features in a logistic regression model for classification. In this task, we utilize the annotated sentences with sentiment labels by *treebank model* introduced in [17]. We report the accuracy in Table I.

The results show that MC and MOMC outperform the baselines as well. Actually, the sentiment of a word is related to the morphological information. For instance, prefix "dis", "un" and "in" have negative meanings. By incorporating morphological information, the morpheme-similar words will group together in vector space. Hence, our better performance may stems from this property.

TABLE II
Performance on Morpheme-rich languages. The numbers in bold mean best performances.

	OSCCA	**MC**	**MOMC**
RG-65-German	58.63	**63.98**	62.36
WS-353-German	59.94	**63.45**	63.21

D. Morpheme-Rich Language Test

In order to measure the ability of our models when applied to some morpheme-rich languages like German, we train OSCCA, MC and MOMC based on the 2009 news German corpus, which is also from the 2013 ACL Workshop on Machine Translation. Then, all word embeddings are tested on word similarity task as well. We utilize Google Translate

[1]http://www.statmt.org/wmt13/translation-task.html
[2]https://msu.edu/∼defores1/gre/roots/gre_rts_afx1.htm
[3]https://github.com/dav/word2vec
[4]https://github.com/paramveerdhillon/swell

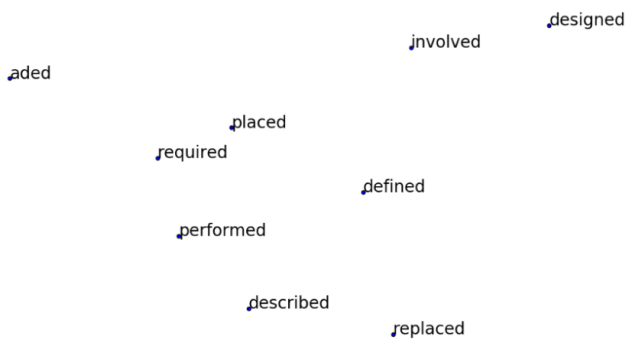


Fig. 1. Words ending with the suffix of "ed" are grouped together in vector space of MOMC.

to translate the golden standard WordSim-353 and RG-65 into German and named them WS-353-German and RG-65-German, respectively.

The results in Table II show that our methods beat OSCCA in a quite much sense. MC outperforms OSCCA more than 5 percentage on RG-65-German and nearly 4 percentage on WS-353-German. In addition, MOMC also performs well on both datasets. Hence, our methods incorporating morpheme information seem to have some positive effects on the quality of word embedding.

E. Word Structure

The word structure of MOMC is illustrated in Fig.1 by using t-SNE. From these figures, we find that the morphological-similar words locate near with each other, which is consistent with our expectation. Moreover, we upload a particular part of the embeddings to the website[5] providing word vector evaluation service by [18]. The result in Fig.2 shows that some male and female related words have exciting semantic structures. For example, in the top left of this picture, tokens of "he", "his", "she" and "her" sit nearby. Furthermore, it also indicates some analogy meaning that the vector from "her" pointing to "his" is parallel to that from "she" pointing to "he". Nevertheless, there are also some weaknesses in Fig.2. For instance, words "mother" and "mom" are semantic-related words but far from each other. As we all know, the quality of word embeddings have a strong connection with the corpus. Obviously, we can not capture all patterns because of some outliers in corpus, which can do harm to the quality of word embeddings.

F. Parameter analysis

In our models, there are several parameters related to the quality of the word embeddings including token size, context window size and the dimension of embedding. We are going to do parameter analysis based on the three parameters in the following parts.

[5]http://www.wordvectors.org/index.php

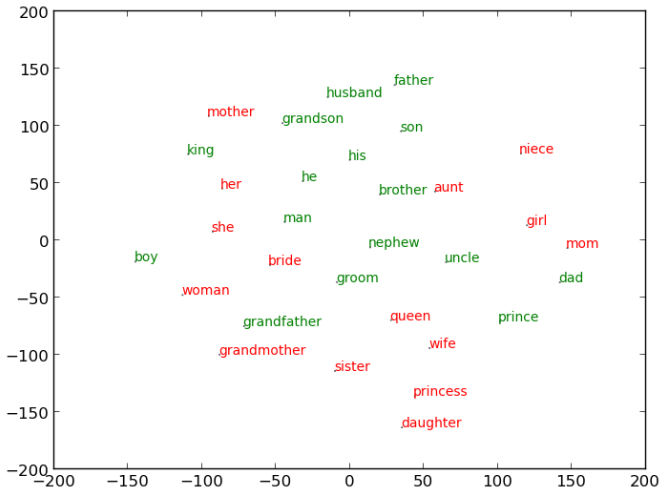


Fig. 2. The position of male and female related words in vector space of MOMC.

1) Effect of token size: In order to analyze the effect of token size, OSCCA, MOMC and MC are trained on the one-fifth, two-fifth, three-fifth, four-fifth and five-fifth of the corpus we mentioned before. All word embeddings are tested on the word similarity by using the golden standard Wordsim-353. The results are illustrated in Fig.3.

In Fig.3, the performance of all the word embeddings shows an obvious ascending tendency, which means larger corpus will generate better word embedding. At the beginning, MC and MOMC perform worse than OSCCA. However, with the increasing of token size, MC and MOMC outperform OSCCA and have a more stable increased tendency.

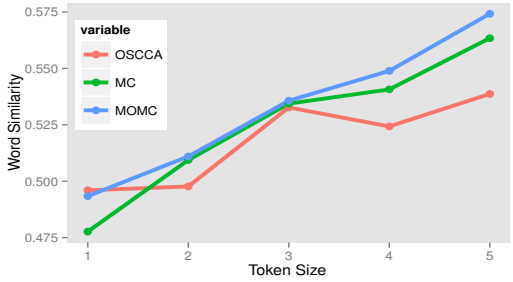


Fig. 3. Effect of token size based on word similarity test by using Wordsim-353.

2) Effect of Context Window Size: In this part, the context window size ranges from 1 to 5. All models are evaluated on the word similarity task by using Wordsim-353 as well. The result is illustrated in Fig.4.

From Fig.4, an ascending tendency is very clear accompanying with the changing of window size, which accords with our expectation. Larger window size means more semantic information. It seems that MC and MOMC are more sensitive to window size. At beginning, MC and MOMC perform worse than OSCCA. However, they outperform OSCCA on larger window size. We can not roughly conclude that larger window size means better performance because we don't test the

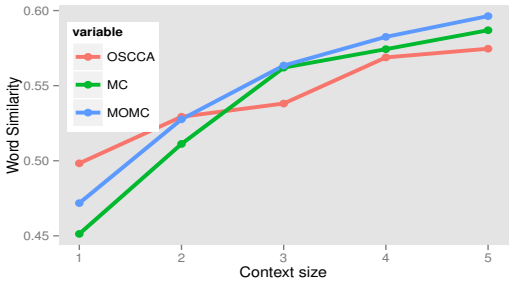


Fig. 4. Effect of context size based on word similarity test by using Wordsim-353.

models by setting some extreme condition due to the limitation of computing resource. In small scope, it is concluded that larger window size generates better word embedding.

3) Effect of Word Embedding Dimension: The word embeddings of OSCCA, MOMC and MC are trained by dynamically setting the dimension increasing from 50 to 250 step by 50. The results on word similarity are shown in Fig.5.

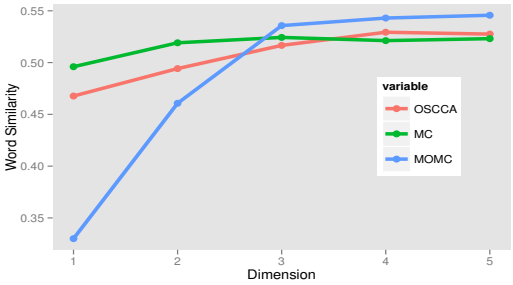


Fig. 5. Effect of dimension based on word similarity test by using Wordsim-353.

In Fig.5, all models follow a tendency of ascending before and stable later. It seems that MOMC is very sensitive to dimension and starts with a low position. An empirical explanation is given to illustrate this phenomenon. From anterior section, it is easy to find that the matrices used in MOMC for random SVD decomposition are much larger than other models which means if the dimension is set too low, we may lose more information than other models when the random SVD is conducted. It convinces us that the dimension of MOMC should be set larger than other models. From the figure, it is fine to choose dimension of word embedding from 200 to 250.

V. Conclusion

Traditional word embedding models neglect meaningful internal semantic structures such as morphemes when extracting semantic and syntactic information from large corpus. To address this problem, we propose two models including MOMC and MC by exploiting the morphological information based on spectral methods. In MOMC, the morphological information is viewed as a unit feature. On the contrary, the morphological information is viewed as a global feature and utilized to supplement the context information during the training process of MC. The experiments' results on word similarity and sentiment classification show that MOMC and MC outperform the baselines on both tasks. The morpheme-rich language test also demonstrates the effectiveness of MOMC and MC, which outperform OSCCA to a great extent. In summary, both semantic-similar words and morphological-similar words have a trend to group together in vector space of MOMC and MC. The property can not only improve the semantic similarity but also elevate morphological similarity of word embeddings.

References

[1] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1, no. 1.

[2] Y. Liu, Z. Liu, T.-S. Chua, and M. Sun, "Topical word embeddings," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015, pp. 2418–2424.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[4] P. S. Dhillon, D. P. Foster, and L. H. Ungar, "Eigenwords: Spectral word embeddings," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 3035–3078, 2015.

[5] X. Chen, L. Xu, Z. Liu, M. Sun, and H. Luan, "Joint learning of character and word embeddings," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2015, pp. 1236–1242.

[6] J. Xu, J. Liu, L. Zhang, Z. Li, and H. Chen, "Improve chinese word embeddings by exploiting internal structure," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1041–1050.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.

[8] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[9] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," *arXiv preprint arXiv:1508.06615*, 2015.

[10] H. Rubenstein and J. B. Goodenough, "Contextual correlates of synonymy," *Communications of the ACM*, vol. 8, no. 10, pp. 627–633, 1965.

[11] G. Halawi, G. Dror, E. Gabrilovich, and Y. Koren, "Large-scale learning of word relatedness with constraints," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2012, pp. 1406–1414.

[12] T. Luong, R. Socher, and C. D. Manning, "Better word representations with recursive neural networks for morphology." in *Proceedings of the SIGNLL Conference on Computational Natural Language Learning*, 2013, pp. 104–113.

[13] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," in *Proceedings of the International Conference on World Wide Web*. ACM, 2001, pp. 406–414.

[14] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa, "A study on similarity and relatedness using distributional and wordnet-based approaches," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2009, pp. 19–27.

[15] E. Bruni, N.-K. Tran, and M. Baroni, "Multimodal distributional semantics," *Journal of Artificial Intelligence Research*, vol. 49, no. 1-47, 2014.

[16] M. Faruqui and C. Dyer, "Non-distributional word vector representations," *arXiv preprint arXiv:1506.05230*, 2015.

[17] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, vol. 1631, 2013, p. 1642.

[18] M. Faruqui and C. Dyer, "Community evaluation and exchange of word vectors at wordvectors.org," in *Proceedings of the Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014, pp. 19–24.

A Lightweight Approach for Evaluating Sufficiency of Ontologies

Lalit Mohan S, Gollapudi VRJ Sai Prasad, Sridhar Chimalakonda, Y. Raghu Reddy and Venkatesh Choppella
Software Engineering Research Center, IIIT Hyderabad, Hyderabad, India

Abstract

[1]*Ontologies have emerged as a common way of representing knowledge. Recently, people with minimal domain background or ontology engineering are developing ontologies, leading to a corpus of informal and under-evaluated ontologies. Existing ontology evaluation approaches require rigorous application of formal methods and knowledge of domain experts that can be cumbersome or tedious. We propose a lightweight approach for evaluating sufficiency of ontologies based on Natural Language Processing techniques. The approach consists of verifying the extent of coverage of concepts and relationships of ontologies against words in domain corpus. As a case study, we applied our approach to evaluate sufficiency of ontology in two example domains - Education (Curriculum) and Security (Phishing). We show that our approach yields promising results, is less effort intensive and is comparable with existing evaluation methods.*

Keywords : Ontology; Ontology Evaluation; Sufficiency

1 Introduction

An ontology is essentially a shared understanding, a unifying framework, a world view of a domain of interest. Ontologies can be about any topic of interest, and as they can be readily merged and made into hybrid structures, it is quite possible that the ontologies can be large. Ontologies are considered significant and reusable as they contained core knowledge structures that require rigor for both development and evaluation. To keep rigor, multiple parameters are checked and detailed criteria is considered for evaluation of ontologies by various researchers [3], [9]. The criteria listed by Vrandevic [16] contains accuracy, adaptability, clarity, completeness, computational efficiency, conciseness, consistency, and other parameters for evaluation. The emergence of semantic web has triggered a need to connect a multitude of web applications from various domains, share and exchange knowledge between them. Several ontology repositories like Protege Ontology Library[2], Linked open vocabularies[3] and search engines like Swoogle[4] and OntoSearch[5] have emerged as a way to access these ontologies. From a utility point of view, software engineers have been using them in their applications for structuring knowledge, sharing a common understanding, explicitly surfacing a given perspective, enabling interaction, navigation, etc. This extensive growth in the use of ontologies poses a critical need to evaluate the quality of ontologies.

Today, informal, loosely defined ontologies have become quite prolific. One of the plausible reason being that ontologies are developed by people with minimal background in ontological engineering, thus making it important to assess the completeness of such ontologies. Completeness is defined as 'all that is supposed to be in the ontology is explicitly stated in it, or can be inferred' [3]. Completeness [16] can be measured from various perspectives: with regards to the language, domain, applications requirements, etc. We are interested in domain and application requirements as it applies to both the goals of the software developer as well as its coverage of the domain the ontology is representing. For an ontology to be complete for a domain, it is necessary for it to represent adequate portion of the domain. However, domain completeness of an ontology cannot be checked as only some of the real world knowledge is available or aspects in real world change over a period of time. We measure completeness as the degree of coverage of real world situations available in the form of web documents. Adopting real world coverage measure for completeness, we introduce **Sufficiency** as a means to measure completeness of the loose and informal ontologies. Our definition of Sufficiency is 'the adequate coverage of specific ontology concepts and relationships for a domain corpus'. The domain corpus would be considered as adequate if the newness of obtained / extracted words tapers. For simple and small domain ontologies, the mechanisms for evaluation, especially for evaluating completeness seems to be under-represented. In our research, we are interested in the problem of evaluating sufficiency of weak, loosely defined domain ontologies.

[1]DOI: 10.18293/SEKE2017-185

[2]http://protegewiki.stanford.edu
[3]https://datahub.io/dataset
[4]http://swoogle.umbc.edu/
[5]http://www.ontosearch.com/

2. Literature Survey

Broadly, the approaches for Ontology evaluation can be classified as (i) manual, mainly driven by human interventions, either experts or users (ii) automated approaches and (iii) semi-automated approaches that fall in between. One way of classification uses black box strategies, which is primarily used from end user perspective or when ontologies are not available during construction, grey box strategies are applied throughout the life cycle of ontologies [4]. A classification by Brank et al. [1] is based on two dimensions (i) type of approach (comparison against a gold standard, application or task-based evaluation, user based evaluation, and data-driven evaluation) and (ii) level of evaluation (lexical, vocabulary, or data layer; hierarchy, taxonomy; other semantic relations; context, application; structure, architecture, design). Ren et al. [11] suggested axiomatic and formalization of competency questions for ontology evaluation. Hlomani and Stacey [6] defined ontology evaluation as verification and validation. However, modeling ontologies using first order logic and formal techniques are daunting tasks that might not be feasible in the case of simple ontologies, which is the focus of this paper. While OntoClean's approach to use formal notions from philosophy such as essence, rigidity, identity and unity for ontology correctness might not be directly relevant for our case, they emphasize the need for validation of ontological adequacy [5].
There are several lines of research that focused on metrics for ontology evaluation. For example, EvaLexon [14], assessed triples mined for text and calculated precision, accuracy and recall values for a domain. The approach had 95% confidence level for 60% coverage. Samir et al. [15] in OntoQA used schema metrics and instance metrics to evaluate ontologies and knowledge bases. They state that "goodness" or the "validity" of an ontology vary between different users or different domains, making it subjective. Astrid et al. [2] extended software product quality SQuaRE, ISO/IEC 25000:2005 to establish OQuaRE framework. The evaluation includes structural, Functional adequacy, Reliability, Performance efficiency, Operability,etc. Gomez et al. [7] proposed OntoMetric with 129 characteristics across 5 dimensions (Tools, Language, Content, Methodology and Costs) for evaluating ontology. Sabou et al. [12] states ontology evaluation is core to ontology selection and have a well laid process for evaluating large scale web based applications. Most of these metric based approaches require extensive information on specific properties of the ontologies that are generally not available for simple ontologies. The ontology coverage check method proposed by Pammer et al. [10] starts with basic domain terms coverage and extends to axioms but their method is focused on individuals with a validity threat that individuals for ontologies are generally not available. Noy et al. [8] suggested using ontology search criteria of the user for evaluating the completeness of ontology. We see that search criteria is an important aspect of evaluation, which we also use in our method but based on domain than on users or specific contexts.

3. Proposed Method

Our intention is to evaluate the sufficiency of a given onotolgy, which has been developed for a particular purpose, against a given domain. This requires us to, identify the test corpus from the domain which is adequate for our evaluation and check for the coverage of ontology in the selected test corpus of the domain.

3.1. Collecting Sufficient Test Corpus

We wish to identify the test corpus of the domain that should be used in our completeness evaluation. The choice of which specific document to consider as corpus is related to the purpose/goal of the ontology. We make an assumption that both the goal and the access to real-to-life test corpus is available. Based on this, we suggest that the type of corpus and the search strings for obtaining corpus should be driven by goals, set for the ontology. The quantity of the test corpus that needs to be considered can indeed vary. To contain this, we bring in the notion of adequate domain corpus. The process for collecting adequate quantity of test corpus is : we select a document, search for unique words in it and count them. If the next document contains more than $Suf\%$ of new words, we add them to the list of unique words and then continue with next document, else, we stop the process. We believe that, after some point, the corpus of words stops being significantly unique. We cut off at $Suf\%$ difference, an arbitrary number and can be changed. The trade-off is that the smaller this number, the more test documents are needed to feed the system. The result of this step is to conclude on the quantity of documents (*SDC* - Sufficient Coverage) that is sufficient for checking our coverage.

3.2. Checking for Coverage

The *SDC* provides sufficient corpus for evaluating the coverage of a specimen ontology. Each individual document of the domain corpus within the set of *SDC* is used for evaluating ontology Concepts, Concepts + Relationships, and Concepts + relationships + Concepts coverage. Individual Concepts label or a Relationship label are represented by C. Concepts + Relationships by R, this R is more restrictive than C because it defines a Concept and potential Relationships of the Concept. For Tuple, Concept + Relationship + Concept is represented as T, this is most descriptive as it contains various destination concepts.

3.2.1 Step 1: Identify Test Ontologies

For our evaluation purposes, we may either have a test ontology or we may need to get one from a ontology repository. For obtaining a preexisting one, we suggest selecting our ontology from a set of at least 3 possible alternatives. The reason we suggest 3 is that if for a domain if we have less than 3, then there is no question of selection and evaluation. The choice is self evident.

3.2.2 Step 2: Extract Labels

A list of concepts and relationships of an ontology give a rough idea of the overall scope and capability of that specimen ontology. Extract and create a list of concept and relationship (C, R, and T) for each of the specimen ontology from a OBO-XML, OWL RDF/XML format. For small, weak, loosely defined ontologies, number of nodes and edges are not expected to be high in number, so we expect this process can be automatic or manual and simple. At the end of this step, there should be three structures for each specimen ontology: One, a list containing clusters of words for each ontology.

$$\{O_1, O_2, ...O_n\}$$

This is C and it should contain all the labels (Concepts and Relationships) in the ontology. Two, A two dimensional array highlighting Concept + Relationships for each ontology.

$$\{O\{c\}\{r\}_1, O\{c\}\{r\}_2, ...O\{c\}\{r\}_n\}$$

Three, L is same as Two but with consideration of Lemmas (grouping of different inflected words such as teach for teaches, taught, etc.) in the corpus. Four, a three dimensional array for each ontology showcasing the Tuples present in the ontology.

$$\{O\{c\}\{r\}\{c\}_1, O\{c\}\{r\}\{c\}_2, ...O\{c\}\{r\}\{c\}_n\}$$

3.2.3 Step 3: Obtain Synonyms

In our work, the implication of using words is that we may be only looking for exact string matches and not consider either common concepts or related words as same. So, if a 'student' is the search word, then it will not match with either a 'pupil' or a 'participant'. There are existing Natural Language Processing (NLP) techniques including Hypernyms, Synonyms to cluster similar words together. On comparison with Synonyms, the coverage boosted between ontologies (Concept and Relationship words) and sample web document sizes. This suggests that a Synonym will effect all data similarly and will not enhance one ontology over the other. Due to this common impact, we rejected usage of Synonyms or other similarity techniques.

3.2.4 Step 4: Identify Test Corpus

Each of the document identified as part of *SDC* and also the aggregation of the text in the documents is part of the test corpus. If the test corpus is hard to identify or define, then chances are that the ontology is no more simple or loosely defined. For such cases, the process needs to be more rigorous and systematic as detailed in literature survey.

3.2.5 Step 5: Pre-process

After selecting test corpus, the content needs to be preprocessed. Pre-processing involves (i) Extract text - in some cases this could be grabbing text from websites or from PDF documents (ii) Ensure that appropriate text in images, tables, audio/video (subtitles) is accessible for extraction while removing non-textual elements (iii) Perform Part-of-Speech (POS) tagging to list Nouns and Verbs in the document (iv) Remove unwanted repeat / stop words like 'and', 'but', 'if' etc. These can be considered as prepositions, conjunctions and other common English words that may not be relevant to domain ontology. The intent of these steps is to ensure that the test corpus is machine ready for evaluation.

3.2.6 Step 6: Collect Unique Words

The aggregate text file content from pre-processing step is processed for extracting the list of most frequently used words. Any text analysis or information retrieval library or online tool that serves this purpose can be used for gathering a bag of words.

3.2.7 Step 7: Compare

This is done by comparing the list of C, R, L and T words of the specimen ontology with all the list of unique words extracted from each document of *SDC* set to determine sufficiency. For comparison, we are attempting to string match the label (of a Concept or a Relationship) and also match the lemmas of the text. The reason for inclusion of lemmas is because a test corpus is always more grounded in instances, whereas a ontology is typically more abstract and at a higher level. As we intend to keep the matching algorithm lightweight, we are not proposing rigorous NLP techniques such as identification of Hypernym, Hyponym, Bi-grams, etc. or any similarity algorithms such as Latent Semantic Analysis and Word2Vec (trained on 100 Billion words). In our work, to check for degree to which an ith ontology is sufficient, we apply this equation

$$(O_i MWFC/OWFC) = O_i \text{Sufficiency (SC)} \qquad (1)$$

Where O_i represents the ith ontology, $MWFC$ represents the frequency count of the matching words, and $OWFC$ represents the total word frequency count of the ontology for

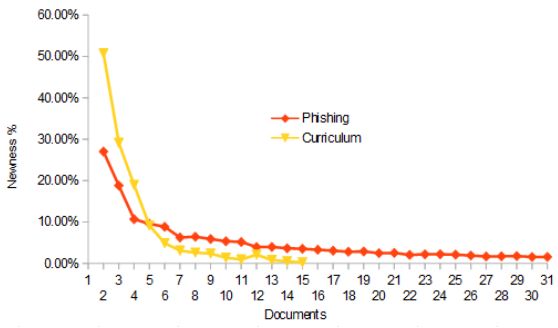


Figure 1. Sufficient Domain Coverage

a given corpus. To check and evaluate something as sufficient, we first take the words from the ontology and string search them within the corpus for C, R, L and T. Once the ontology related words are found in the corpus, then they are identified and their frequency counts are obtained. The total frequency count of the matched words is checked against the overall frequency count of all words. This ratio is said to be the measure of Sufficiency SC.

4. Evaluation of the Approach

We now evaluate our approach by applying it to the various publicly available Curriculum and Phishing ontologies. Our goal is to select an ontology with the highest degree of Sufficiency for a hypothetical project.

4.1. Selection of Test Corpus

To gather test corpus, we used popular search engines like Google, Bing, Yahoo, Yandex and Baidu with search string as 'Curriculum' and 'Phishing'. From the search results, a union collection of documents are gathered as our test corpus [13]. The text in each of the documents is concatenated into a single document and the newness of words is identified. The Sufficient Domain Coverage figure 1 shows that the newness tapers after 12 documents for Curriculum and after 25 documents for Phishing. The site specific words such as Contact address, Organization name, creative English by writers, etc. are probable reasons for the newness value not being zero.

4.2. Pre-processing of Corpus

Pre-processing involves conversion of PDF to Text mode for some cases, scrapping content (removal of html tags, css, images, etc.), removal of stop words, etc. Removal of stop words and word frequency count was done by a Java application[6] that we developed. Along with this, we also performed POS tagging to identify the list of Nouns and Verbs using Stanford NLTK. The count [13] of stop words, unique words and the newness for Curriculum and Phishing Ontology is obtained.

4.3. Handling of Ontologies

A web search on the word Curriculum and Phishing was used to identify three Ontologies including Ontosearch and Swoogle. After earmarking the specimen ontologies, the next task of extracting labels from each of ontologies separately was done.

4.4. Word Comparison

After extracting labels from ontologies, each of those labels were string searched for potential matches in the words list of the test corpus. Whenever there was a match, the matching frequency was obtained and aggregated. This gave us O_i MWFC or the matching word frequency count of the ith ontology.

4.5. Checking for Completeness

For the ontology, the last task is to calculate the sufficiency as degree of a completeness score. This score is calculated (as given by equation 1) by dividing O_i MWFC by the OWFC value. See in [13], D1 through D12 indicates documents identified as part of the corpus and C12 indicates the combination of all the documents. Sample Ontologies for Curriculum (O_1^c, O_2^c and O_3^c) and Phishing (O_1^p, O_2^p and O_3^p) are used for determining ISC - Individual Sufficient Completeness (concepts and relationships). An average of ISC is considered for arriving at Sufficient Completeness SC, however, any other statistical approach can be considered for the calculation.

4.6. Results and Discussions

We tested 12 and 25 sample sets of corpus against 6 (3 of Curriculum and 3 of Phishing) ontologies. As observable in [13], unique words constitute 35-40% of overall word count with some words related to domain being more prevalent [7]. An ontology can be said to be sufficiently complete if, after matching the goals of the ontology, C, R and L extracted from the ontology fully encompass the words of the corpus.

- For Concepts or Relationships C of Curriculum is 64.06% for O_1^c and 61.25% for O_2^c. For Phishing, the score is 67.13% for O_2^p and 51.28% for O_1^p.

[6] https://github.com/lalitsanagavarapu/OntologyEval

[7] http://tinyurl.com/UniqWord

- For Concepts and Relationships R of Curriculum is 60.26% for O_1^c and 24.24% for O_2^c. For Phishing, the scores is 51.75% for O_2^p and 40.17% for O_1^p. The concepts and relationships are compared as Nouns and Verbs after POS tagging the web documents.
- For Concepts and Relationships with application of Lemma L on Curriculum, the score is 73.72% for O_1^c and 70.45% for O_2^c. For Phishing, the score is 59.79% for O_2^p and 45.30% for O_1^p.
- From results, Ontology O_1^c for Curriculum and O_2^p for Phishing stand out as better suited for our application.

Like the check for completeness, coverage of corpus in an ontology too appears to be an audacious goal. Hence, we reject our hypothesis that

$$C \subset O_i$$

. In our evaluation, we considered $Suf\%$ of 2% as the cut-off percentage for newness with 12 documents of Curriculum and 4% (12 documents) for Phishing. The test corpus selection is subjective step in our proposed approach. Hence, we performed the test to see if the concepts and relationships of any one of the earmarked ontologies are present in the R_8 random (include text from the novel Pride & Prejudice; Wiki content on Auto, Health, Sport, Finance, Food, Travel; and a magazine article on 'top technological trends') corpus. The lower Sufficiency Coverage value in the results [13] indicates that most of the ontologies are poorly represented in the random corpus. The combined random corpus content with its 186,360 words and with its 9,801 unique words did not gain much in completeness. The lower numbers for our random sample also indirectly reinforces the other point that the sufficiency value of O_1^c for Curriculum and O_2^p for Phishing is not accidental, but indeed intentional and specific to the ontology. Our approach that is automatic, informal using web documents as domain data as compared to various evaluation approaches. Our approach is similar to OntoQA [15] but has lesser steps and lesser metrics to evaluate ontology.

5. Conclusions and Future Work

There are many techniques to evaluate ontology and most of them appear to be rigorous and tend to target the evaluation of well defined and large ontologies. In such context, we sought and evaluated a lightweight approach for checking sufficiency of smaller ontologies. The approach is simple as it relies on concept and relationship matching and conventional web search techniques. Our evaluation explored veracity of the approach and established the feasibility on two different domains and could extend for other weak and loosely defined ontologies. As a forward plan, we plan to make a tool online instead of running it as a batch process so that other users can leverage it. We also plan to use the domain knowledge available in the web documents including text cohesiveness to evolve ontologies based on identifiable patterns.

References

[1] J. Brank, M. Grobelnik, and D. Mladenic. A survey of ontology evaluation techniques. In *Proceedings of the conference on data mining and data warehouses (SiKDD 2005)*, pages 166–170, 2005.

[2] A. Duque-Ramos, J. T. Fernández-Breis, M. Iniesta, M. Dumontier, M. E. Aranguren, S. Schulz, N. Aussenac-Gilles, and R. Stevens. Evaluation of the oquare framework for ontology quality. *Expert Systems with Applications*, 40(7):2696–2703, 2013.

[3] A. Gómez-Pérez. Ontology evaluation. In *Handbook on ontologies*, pages 251–273. Springer, 2004.

[4] G. Grigonyte. *Building and evaluating domain ontologies: NLP contributions*. Logos Verlag Berlin GmbH, 2010.

[5] N. Guarino and C. A. Welty. An overview of ontoclean. In *Handbook on ontologies*, pages 201–220. Springer, 2009.

[6] H. Hlomani and D. Stacey. Approaches, methods, metrics, measures, and subjectivity in ontology evaluation: A survey. *Semantic Web Journal*, pages 1–5, 2014.

[7] A. Lozano-Tello and A. Gómez-Pérez. Ontometric: A method to choose the appropriate ontology. *Journal of database management*, 2(15):1–18, 2004.

[8] N. F. Noy, P. R. Alexander, R. Harpaz, P. L. Whetzel, R. W. Fergerson, and M. A. Musen. Getting lucky in ontology search: a data-driven evaluation framework for ontology ranking. In *International Semantic Web Conference*, pages 444–459. Springer, 2013.

[9] L. Obrst, W. Ceusters, I. Mani, S. Ray, and B. Smith. The evaluation of ontologies. In *Semantic web*, pages 139–158. Springer, 2007.

[10] V. Pammer, P. Scheir, and S. Lindstaedt. Ontology coverage check: support for evaluation in ontology engineering. In *FOMI 2006. The 2nd workshop: Formal Ontologies Meet Industry*, 2006.

[11] Y. Ren, A. Parvizi, C. Mellish, J. Z. Pan, K. Van Deemter, and R. Stevens. Towards competency question-driven ontology authoring. In *European Semantic Web Conference*, pages 752–767. Springer, 2014.

[12] M. Sabou, V. Lopez, E. Motta, and V. Uren. Ontology selection: Ontology evaluation on the real semantic web. 2006.

[13] L. Sanagavarapu. Ontology - corpus comparison. `http://tinyurl.com/OntoLightWeight`, 2017.

[14] P. Spyns. Evalexon: Assessing triples mined from texts. *STAR*, 2005(09):09, 2005.

[15] S. Tartir, I. B. Arpinar, and A. P. Sheth. Ontological evaluation and validation. In *Theory and applications of ontology: Computer applications*, pages 115–130. Springer, 2010.

[16] D. Vrandečić. Ontology evaluation. In *Handbook on Ontologies*, pages 293–313. Springer, 2009.

Understanding the Impact of Using Ontology Matching Tools for Validating Feature Models with Domain Knowledge

Nada M. Hassan
Arab Academy for Science, Technology and Maritime Transport
Cairo, Egypt
nadamahmoudhassan@aast.edu

Haitham S. Hamza Member, IEEE
Cairo University
Giza, Egypt
hhamza@fci-cu.edu.eg

Yasser M. K. Omar
Arab Academy for Science, Technology and Maritime Transport
Cairo, Egypt
dr_yasser_omar@yahoo.com

[1]*Abstract*— **Feature models (FM) are a way for modeling and describing the product family of specific domain. They are widely used for describing the requirements in the domain engineering as it describes the commonalities and the differences of related products in specific domain. Currently the research in the feature model analysis and the validation of it focus on capturing the inconsistencies of the feature configurations of software systems. However the semantic web had been used for representing the feature models as ontology using OWL DL to use the Description Logic (DL) reasoners in validating the consistency of the feature model configurations, detecting the semantic contradictions or semantic mappings with certain domain is missed. The aim of this research is to detect the semantic mappings between a feature model and a specific domain ontology using the ontology marching tools. In the paper we used the Wireless Sensor Actuator Network (WSAN) feature model for analysis, and the well-known Semantic Sensor Network ontology (SSN) for validation. Two ontology-matching tools are used to map the feature model and the domain ontology, and the results have been compared.**

Keywords—feature model; ontology; description logic; ontology matching; semantic mapping; wireless sensor network

I. INTRODUCTION

Software product lines (SPLS) depend on the idea of producing a family of software products from common features instead of producing them from scratch. The feature model has been firstly introduced in the Feature-Oriented domain analysis (FODA) method as a structure handles the product line members and determining the features that differentiate the software systems in a specific domain [1].

Feature models are most widely used for managing commonalities and variabilities of the features between different systems in a certain domain.

Introducing a new member to SPL, requires feature configurations to define the set of features that describe this member and this configurations are accepted if it doesn't violate the constraints defined by the feature model.The automated analysis of FMs is used to check the validity of the combination of features in a product. This process is mainly carried out in two–phases. "In the first phase, the feature model is converted into a specific representation (e.g. Propositional Logic, Description Logic, etc). Then in the second phase, an off–the–shelf solver or some algorithms are used to analyze the representation of the feature model parameters automatically and provide the result" [2]. Validating the feature model configurations and consistency doesn't reveal the semantic contradictions or semantic mappings with certain domain. This may lead to generate product with illogic features with respect to its domain.

To the best of our Knowledge, none of the existing studies that used the power of DL focused on validating the feature model with certain domain. Detecting the semantic mappings between a feature model and a certain domain may help in the process of validating a feature model with domain knowledge.

In this paper we present how we can use the ontology matching tools to detect the semantic mappings between WSAN FM and SSN ontology. We used the OWL DL to represent the FM, then we tested two ontology matching tools. The paper is organized as follows: sectionII presents the background. Section III provides the related work. Section IV presents a proposed hybrid model applied on a feature model for WSAN [5]. Section V demonstrates the Experimental Results followed by the Conclusion and Future work.

II. BACKGROUND

This section explains the basis of the feature models and the ontology matching techniques.

A. Feature Models

Feature models are used for representing the features of SPL. They have a tree structure. The model starts with root feature followed by sub-features with different relationships with its parent feature.The main types of the FM relations are:

- Mandatory: means the feature must be included into the description of a concept
- Optional: means the existence of the feature is optional
- Alternative: means just one feature from a set of features can be included
- OR: means one feature or more can be included from a set of features.

DOI reference number: 10.18293/SEKE2017-199

B. *Ontology Matching Techniques*

Ontology matching is a complicated process that helps in detecting the semantic correspondences between different ontologies of the same domain, the matching techniques are divided mainly into two types:

- Element-Level technique

This technique is based on obtaining the correspondences between the entities with ignoring the structure of the ontology, this technique is divided into string-based, language-based and linguistic resource as in [14].

- Structure-Level technique

This technique focuses on obtaining the correspondences based on the relations between the entities within the structure of the ontology, this technique is divided into taxonomy mapping, and tree-based mapping as in [14].

III. RELATED WORK

This Section is divided into two parts: Related work of the automated analysis of the Feature Models using DL and representing Feature Model in OWL.

A. *Automated Analysis of Feature Models*

Wang, Hai H., Yuan Fang Li, Jing Sun, Hongyu Zhang, and Jeff Pan were the first to use the DL in feature model analysis to check the feature model consistency with specific configurations. They represent the FM as ontology using OWL DL, then they used DL Racer and FACT++ to check the consistency of the feature configurations automatically, and also they proposed a debugging OWL tool[6].

Zaid, Lamia Abo, Frederic Kleinermann, and Olga De Troyer proposed an ontology framework for feature modeling that consists of an ontology that formally provides the structure of feature models to check the consistency of FM through rule based model, and provide means to integrate segmented feature models[7].

B. *Representing Feature Model in OWL*

As mentioned above, different ways for representing the feature model as ontology had been introduced in [6] and [7].

Tenório, Thyago, Diego Dermeval, and Ig Ibert Bittencourt proposed a generic ontology called OntoSPL represents the structure of the feature model and its relations using OWL classes and properties, so the features can be represented as OWL individuals from this structure, this approach is useful for the dynamic changes at the runtime, although it's not tested on large feature model[8].

Nima Kaviani, Bardia Mohabbati, Dragan Gasevic, and Matthias Finke used the approach that proposed by Wang et al to represent the feature model in ontology to cover non functional requirements, annotate and expand feature models context of ubiquitous environments[9].

As the approach in [6] has been used in other research work, and they tested their approach on a feature model for a large system which contains almost 1000 different features and more than 400 different feature relations, then they applied a debugging process on it. So it is decided to use this approach in the step of representing the feature model in OWL.

IV. PROPOSED HYBIRD MODEL

Our experimental work tested on the WSAN feature model and the SSN ontology, we used two ontology matching tools to detect the mappings between them, Fig.1 shows the steps of our experimental work.

WSAN is composed of large number of sensor nodes and actuators to sense, actuate and communicate to provides specific functionality,While the SSN ontology describes sensors and its properties, capabilities and observations[11].

The experimental work divided into two main parts:

A. *Constructing the ontology from WSAN*

We used the approach in [6], which presents how to use the OWL DL to represent the FM. They define OWL class for each feature in the FM and make them mutually disjoint then using existential restrictions they bind the rule classes that has been created for each feature to the corresponding produced OWL class, each rule class is used to define the constraints over the feature (OWL class).

Table I. contains the DL syntax that used in representing the axioms of the produced ontology (WSAN ontology)

It can be seen from Fig. 2 that the first level of the WSAN FM contains mandatory feature (i.e. Communication) that can be represented in DL as shown below.

$Communication \sqsubseteq T$ $\quad$ $CommunicationRule \sqsubseteq T$

$hasCommunication \sqsubseteq ObjectProperty$

$T \sqsubseteq hasCommunication.Communication$

$CommunicationRule \equiv \exists\ hasCommunication.Communication$

$WSAN \sqsubseteq \exists\ hasCommunication.Communication$

TABLE I. SUMMERY TABLE OF THE DL SYNTEX USED IN THIS PAPER

Notation	Explanation
$\top$	Superclass of all OWL classes
$A \sqsubseteq B$	A is a subclass of B
$A \sqcup B$	Class union
$A \equiv B$	Class equivalence
$\exists = \forall$ P.A	allValuesFrom=someValuesFrom restriction for every instance of this class that has instances of property P, all some of the values of the property are members of class A

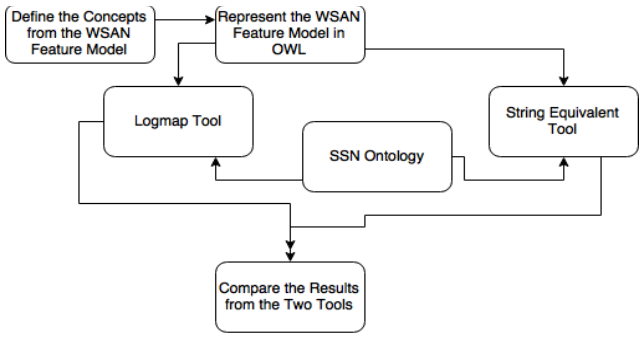


Fig. 1. Block diagram for the steps of the proposed model

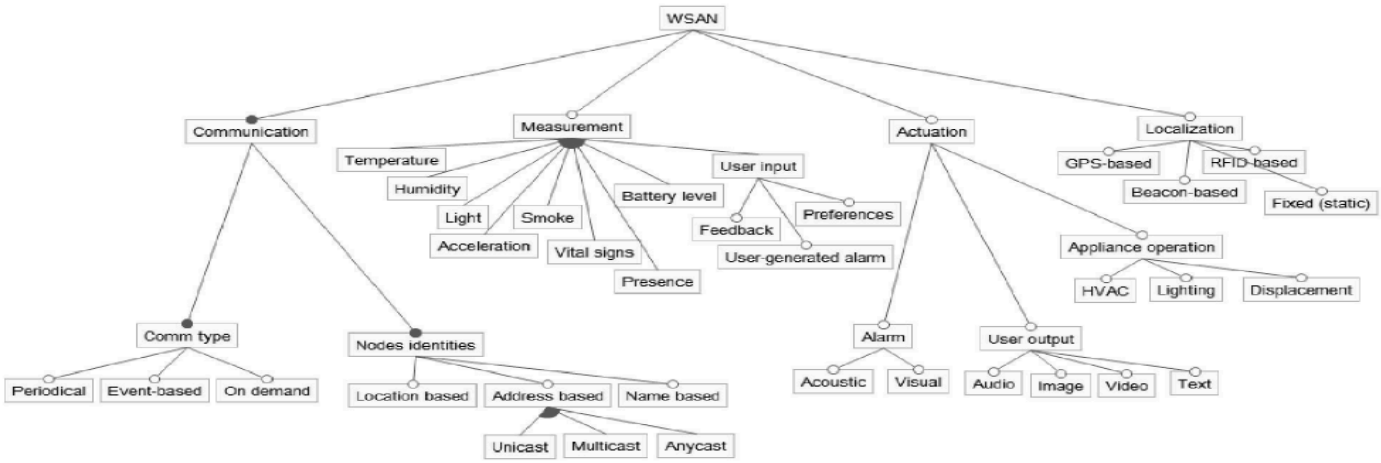


Fig. 2. Feature Model of Wireless Sensor Actuator Network [5]

In Fig.2 the optional features of the first level (e.g. Measurement, etc…) can be represented in DL as shown.

$Measurement \sqsubseteq T$ $MeasurementRule \sqsubseteq T$

$hasMeasurement \sqsubseteq ObjectProperty$

$MeasurementRule \equiv \exists\ hasMeasurement.Measurement$

The Address-based feature has three sub-features with OR relation as shown in Fig.2 that can be represented in DL as shown assuming the Rule classes have been created for each.

$AddressBasedRule \sqsubseteq ((\exists hasUnicast.Unicast) \sqcup (\exists hasMulticast.Multicast) \sqcup (\exists hasAnycast.Anycast))$

The rest of the features of the WSAN feature model have been represented in the same way .

B. Define the mappings between the WSAN produced ontology and SSN ontology

The produced ontology from WSAN FM is used to be mapped with SSN ontology using the ontology matching tools. We used two ontology matching tools (i.e. Logmap, String Equality tool) to generate the mappings.

- Log-Map

The LogMap tool is developed by the University of Oxford. The tool depends on lexically and structurally indexing the input ontologies as the matching process is carried out through five phases[12].

A. Lexical indexation

Logmap identifies for each input ontology the classes labels and its lexical variants using external lexicons (i.e. WordNet, UMLS lexicon).The UMLS lexicon is a set of files that contains many biomedical terminologies, and vocabularies.

B. Structural indexation

The logmap creates the extended class hierarchy through the use of structural heuristics or DL reasoner for the input ontologies.

C. Computation of initial (anchor mappings)

Logmap computes the exact lexical correspondences which are act as initial set of anchor mappings by intersecting the lexical indexes of each input ontology.

D. Mapping repair and discovery

- In the repair process, the Log-Map detects the unsatisfiable classes from merging the two input ontologies using a reasoning algorithm, and then it repairs these undesirable correspondences using a greedy diagnosis algorithm.
- For the discovery phase, according to two sets of semantically related classes new mappings are computed by matching the classes in these two sets using a tool that computes a similarity score.

E. Ontology overlapping estimation

Logmap creates two modules for each ontology, these modules represent the overlapping between O1 and O2. If a correct mapping between O1 and O2 is missed by logmap, when manually looking in these modules you can find it.

- String Equality tool

The String equality tool had been proposed in [10]. This tool is based on string based technique and semantic technique. It works to find the similar classes between input ontology files. The algorithm is based on using synonym file that contains the terms and synonyms of the domain that the input ontologies related to it and also detect the exact mappings.

Finally, the results of the ontology matching tools (i.e. Logmap tool, String Equality tool) were compared and evaluated according to predefined expected results.

V. EXPERIMENTAL RESULTS

We tested two ontology matching tools (i.e. Logmap tool and String equality tool) to map WSAN ontology and the SSN ontology. The expected results from the matching process are shown in TableII, while the comparison between the two tools and its output are shown in TableIII. Followed by Fig3. that shows part of the SSN ontology.

TableIII. shows no classes mapped using logmap, although it generates one object property mapping._The object property "hasLocation" in SSN mapped with "hasLocalization" object property in WSAN but semantically both of these relations connect classes with different semantic meaning and represent different context, also it mapped wrongly the "Measurement capability" in SSN with "Measaurement" in WSAN and both of them describe different context.

TABLE II. EXPECTED RESULTS

Ontology	WSAN	SSN
Mapped Classes	WSAN	System
	Communication_Type	Stimulus
	Communication_Type	Sensor_Input

TABLE III. COMPARISON BETWEEN THE TESTED ONTOLOGY MATCHING TOOLS

Tool	Input	Output	Ontology Matching Technique
Log-Map	File1: WSAN Ontology File2: SSN Ontology	(1) Class mapped (1) Object Property mapped	Based on obtaining initial set of mappings, then it use these mappings to discover new mappings
String Equality Tool	File1: WSAN Ontology File2: SSN Ontology	(2) Class in WSAN mapped with (3) classes in SSN	Based on string based technique, language based method and semantic technique, it uses external file contains the synonyms of the domain

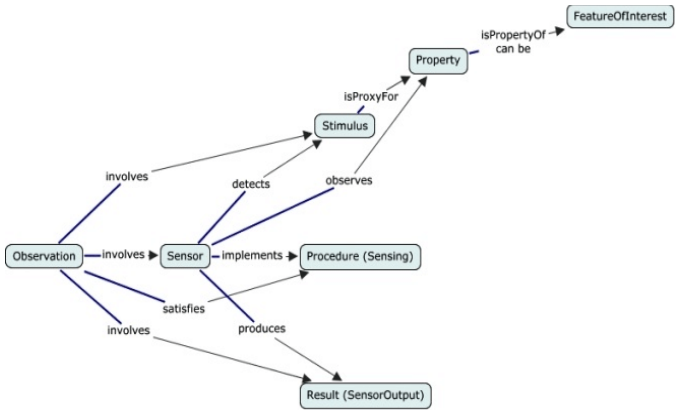


Fig. 2. Part of the Semantic Sensor Network [13]

On the other hand the String equality tool gets two mappings between the input ontologies. The "communicationType" class in WSAN ontology which represents the stimulus that triggers the sensors to work is mapped with "Stimulus" class in Fig.3 and "Sensor_Input" class that are equivelent classes in SSN ontology which represent the stimulus that triggers the sensor, and also it mapped the "WSAN" with "System" in SSN.

According to the expected results in TableII and the output in TableIII, the String Equality tool shows better results than the Logmap tool. LogMap doesn't produce all the mappings in the nonbiomedical domains.

The results of the mapping can be considered as a resonable results, as the SSN ontology describes the sensors and its observations, while the WSAN describes the Network of connected wireless sensors and actuators.

CONCLUSION AND FUTURE WORK

This paper presented a comparison between two different matching tools (i.e. LogMap tool , String Equality tool) that have been used for detecting the semantic mappings between the WSAN FM and SSN ontology as a domain knowledge using the ontology matching tools. The OWL DL is used for representing the FM as ontology using wang et al approach.

We can conclude from the results that the generated mappings can help in determining to what extent a feature model matches a certain domain, as in our experiement the WSAN FM wasn't completely matched with the SSN Ontology.The results also shows, that the String Equality tool shows better results than the logmap in our experiment. Therefore having lexicon for the terminologies of the domain that the maping is done over it, gives better results.

In the future work, it's intended to extend this work and test it on other domains to complete the process of the validation and draw recommendations for the features that can be added to the feature model from the ontology.

REFERENCES

[1] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson, "Feature–Oriented Domain Analysis (FODA) Feasibility Study". Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, November 1990.

[2] Benavides, David, Sergio Segura, and Antonio Ruiz-Cortés. "Automated analysis of feature models 20 years later: A literature review." Information Systems 35, no. 6 (2010), pp. 615-636.

[3] Kim, Chang Hwan Peter. On the relationship between feature models and ontologies. University of Waterloo, 2006.

[4] Otero-Cerdeira, Lorena, Francisco J. Rodríguez-Martínez, and Alma Gómez-Rodríguez. "Ontology matching: A literature review." Expert Systems with Applications 42, no. 2 (2015), pp. 949-971.

[5] Ortiz, Óscar, Ana Belén García, Rafael Capilla, Jan Bosch, and Mike Hinchey. "Runtime variability for dynamic reconfiguration in wireless sensor network product lines." In Proceedings of the 16th International Software Product Line Conference-Volume 2, pp. 143-150. ACM, 2012.

[6] Wang, Hai H., Yuan Fang Li, Jing Sun, Hongyu Zhang, and Jeff Pan. "Verifying feature models using OWL." Web Semantics: Science, Services and Agents on the World Wide Web 5, no. 2 (2007), pp. 117-129.

[7] Zaid, Lamia Abo, Frederic Kleinermann, and Olga De Troyer. "Applying semantic web technology to feature modeling." In Proceedings of the 2009 ACM symposium on Applied Computing, ACM 2009, pp. 1252-1256.

[8] Tenório, Thyago, Diego Dermeval, and Ig Ibert Bittencourt. "On the use of ontology for dynamic reconfiguring software product line products." In Proceedings of the ninth international conference on software engineering advances, 2014, pp. 545-550.

[9] Kaviani, Nima, et al. "Semantic annotations of feature models for dynamic product configuration in ubiquitous environments." 4th International Workshop on Semantic Web Enabled Software Engineering. at 7th International Semantic Web Conference. 2008.

[10] Gawich, M., et al. "Alternative Approaches for Ontology Matching." International Journal of Computer Applications 49.18 (2012).

[11] Compton, Michael, Payam Barnaghi, Luis Bermudez, RaúL GarcíA-Castro, Oscar Corcho, Simon Cox, John Graybeal et al. "The SSN ontology of the W3C semantic sensor network incubator group." Web semantics: science, services and agents on the World Wide Web 17 (2012), pp. 25-32.

[12] Jiménez-Ruiz, Ernesto, and Bernardo Cuenca Grau. "Logmap: Logic-based and scalable ontology matching." International Semantic Web Conference. Springer Berlin Heidelberg, 2011.

[13] Available:https://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/. [Accessed:10-1-2016]

[14] Hooi, Yew Kwang, M. Fadzil Hassan, and Azmi M. Shariff. "A survey on ontology mapping techniques." In Advances in Computer Science and its Applications, pp. 829-836. Springer Berlin Heidelberg, 2014

Towards a Model-Driven Platform for Evidence based Public Health Policy Making

Marios Prasinos, George Spanoudakis
City, University of London
London, United Kingdom
{Marios.Prasinos.1@city.ac.uk,
G.E.Spanoudakis@city.ac.uk}

Dimitrios Koutsouris
Biomedical Engineering Laboratory,
National Technical University of Athens
Athens, Greece
dkoutsou@biomed.ntua.gr

***Abstract*— The effective management of various health conditions depends on and requires appropriate public health policies (PHP). Such policies are important for several aspects of healthcare provision, including: (a) screening for prevention of disease; (b) early diagnosis and treatment; (c) long-term management of chronic diseases and disabilities; and (d) setting-up standards. Although it is widely recognised that the PHP life cycle (i.e., the analysis, action plan design, execution, monitoring and evaluation of public health policies) should be evidenced based, current support for it is mainly in the form of guidelines, and is not supported by data analytics and decision making tools tailored to it. In this paper, we present a novel model driven approach to PHP life cycle management and an integrated platform for realising this life cycle. Our approach is based on PHP decision making models. Such models steer the PHP decision making process by defining the data that need to be collected and the ways in which these data should be analysed in order to produce the evidence required for PHP making. Our work is part of a new research programme on public health policy making for the management of hearing loss, called EVOTION, that is funded by the European Union.**

Keywords — public health policy; model driven data analytics; ontologies

I. INTRODUCTION

The effective management of various health conditions depends on and requires appropriate public health policies (PHP) as it has been acknowledged by the World Health Organisation [1], governmental institutions and patient associations [2]. Public health policy affects the affordability and, hence, access to health care services (e.g., health check-ups, health care device adjustments, provision of related rehabilitation services), medication and supportive devices. PHP may also have a significant effect on: (a) screening for prevention of disease; (b) early diagnosis and treatment; (c) long-term management of chronic diseases and disabilities; and (d) setting-up standards.

It is widely recognised that the PHP life cycle (i.e., the analysis, action plan design, execution, monitoring and evaluation of public health policies) should be evidenced based. This is because the management of health conditions and their consequences at a public health policy making level can benefit from the analysis of heterogeneous data, including health care device usage (if applicable), physiological, cognitive, clinical and medication, personal, behavioural, life style data, occupational and environmental data. The analysis of these types of data using big data analytics techniques can enable the investigation of whether particular health conditions have comorbidities and reveal contextual factors, social, behavioural and economic, life cycle and other factors affecting them. The outcomes of such analysis can enable the stratification of related risks and effects to the patients, and – through correlation with other economic, social and physical constraints – help developing a holistic systemic perspective of interventions regarding the management of health conditions and the broader support, social and occupational inclusion and well-being of the patients. Evidence arising from big data analytics (BDA) can also help exploring missing, under or over-estimated value of specific medical interventions and analysing their effectiveness (i.e., understanding the trade-offs between their cost and benefits). Despite such potential benefits, current support for PHP making is mainly in the form of guidelines, and is not supported by data analytics and decision making tools based on such analytics.

In this paper, we present a novel model driven approach to PHP life cycle management and an integrated platform for realising this life cycle. Our approach is based on PHP decision making (PHPDM) models. Such models steer the PHP making process by defining the data that need to be collected and the ways in which they should be analysed in order to produce the evidence required for PHP making. Our work is part of a new research programme on public health policy making for the management of hearing loss, called EVOTION, that is funded by the European Union. On-going reforms of PHP in this area related to the management of different types of hearing loss and the spark of social debate that they have caused demonstrate the importance of PHP making in this area (see, for example: https://www.actiononhearingloss.org.uk/get-involved/campaign.aspx).

The rest of this paper is structured as follows. Section II describes related work. Section III provides an overview of our approach and the architecture of platform that we are developing to realise it. Section IV presents the ontology based scheme for specifying PHPDM models. Section V presents an example PHPDM model and how it can be executed to realise our approach. Finally, Section VI presents concluding remarks and directions for future work.

DOI reference number: 10.18293/SEKE2017-035

II. Related Work

A recent review of big data applications in biomedical research and health care [3] has pointed out several applications, including, for example: real-time risk monitoring, using online social media combined with epidemiological information as a data source for facilitating public health surveillance [4] and ambulatory cardiovascular care [5].

The opportunities, which arise from using big data analytics to reduce the costs of health care [6] are also unprecedented. Examples of healthcare services that can benefit in this respect include high-cost patients, readmissions, triage, decompensation (when a patient's condition worsens), adverse events, and treatment optimisation for diseases affecting multiple organs [6]. The use of big data analytics in healthcare may also enable the generation and dissemination of new knowledge, the translation of personalised medicine initiatives into clinical practice and the transformation of healthcare by delivering information directly to patients, empowering them to play a more active role [7].

According to a systematic review of studies of decision-making by health care managers and policy-makers, researchers could better inform health care management and policy-making by making several changes to how they produce and update systematic reviews and by adapting existing reviews that are relevant to local health care issues [8]. Also, research in this area becomes important as policy-makers and managers increasingly require access to high-quality evidence syntheses that include research and non-research based evidence, in the form of qualitative and/or quantitative research findings [9]. `Recent applications of big data research in health policy making include breast screening decision making [10] and public health strategies for alcohol harm reduction in the UK [11].

A key to success in learning from big health care data is to remain focused on gaining actionable insights into the best ways to treat the patients in the care system that generated the data [12].

III. Approach

To realise the approach that we outlined in Sect. I, we are developing a platform (referred to as the *EVOTION platform* in the following) whose overall architecture is shown in Figure 1. This platform uses different types of data to inform the PHPDM (PHPDM) process. Such data include: (a) retrospective and prospective patient data, including medical, clinical and medication, personal and occupational data; (b) prospective real time patient data including medical devices usage, cognitive, behavioural and life style, and environment data (e.g., location of patient, noise environment); and (c) dynamic web and social media data (e.g., feedback on proposed or implemented policies that may be useful in predicting or evaluating the perceptions of the public about PHPs).

The operation of the EVOTION platform is driven by PHPDM models. These models specify:

(i) The generic issues that need to be addressed by PHPs and the alternative decisions that may be made to address them.

(ii) The evidence that can support or provide counter indicators for decisions. Evidence may be related to a wide spectrum of factors. Considering the use of medical devices, for example, evidence can be collected to explore whether the difficulties faced by different types of medical device users depend on their condition, their cognitive capabilities, their life style and behaviour, other comorbidities that they may have and/or their overall compliance with medical device usage guidelines given to them by clinicians. Evidence may also be required to explore whether such difficulties can be alleviated by the number of follow up treatments, the time periods between such treatments etc.

(iii) The big data analytic (BDA) processes (e.g., the specific types of statistical analysis or data mining analysis) that should be followed for collecting and analysing the evidence.

(iv) The criteria that should be used to determine if the available evidence is sufficient for making decisions. Criteria determine the extent of the evidence that would be deemed sufficient for supporting a decision, and thresholds that would make the evidence conclusive. Criteria may, for example, determine the combination of the factors referred to in (ii) above that would be a good predictor of the difficulties faced by medical device users.

(v) The processes to be followed for making specific types of health policies. Such processes may, for example, determine who are the stakeholders whose views should be considered and recorded prior to reaching a decision, who has responsibility for making the final decision, and whether a decision should be continually or periodically reviewed upon the acquisition of new evidence.

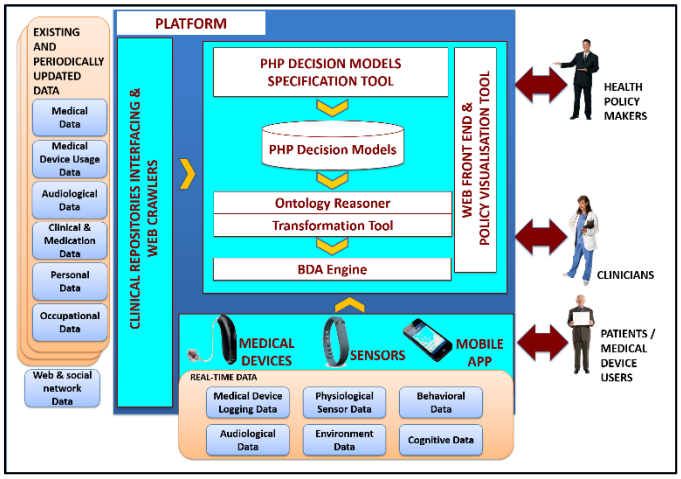


Figure 1 The Overall Evidence Based PHP Decision Making Platform

Our view is that PHPDM models are essential for realising evidence based, scalable, fully dynamic, repeatable and accountable policy making. This is because PHPDM models covering the aspects identified above could:

- be automatically transformed into executable BDA processes whose execution would provide the basic evidence required for making a decision and exploring its consequences;
- drive the collaborative stakeholder decision making processes; and

- provide a structure for organising the alternatives, arguments and rationale for making decisions in a way that makes them traceable and accountable.

PHPDM models can be: (a) repeatedly executed in the same or different policy making settings (e.g., for making policy on the very same issues in different regions); and (b) specified parametrically to make their customisation easy in case that this would be required in different policy making settings.

As shown in Figure 1, the EVOTION platform incorporates a tool supporting the specification of PHPDM models into some high-level language, and their verification and transformation into executable BDA processes that would be passed as input to the BDA engine to drive its execution and generate the evidence required for PHPDM. Also, to enable the data collection processes and policy-making process, the EVOTION platform incorporates and integrates:

(a) Existing repositories of medical data.
(b) Enhanced medical devices enabling the capture and provision of medical device usage related data (e.g., rating of device ease or difficulty of use in different listening conditions, frequency and type adjustments of controls).
(c) Sensors supporting the collection real time contextual patient physiological data (e.g., heart rate, blood pressure, skin conductance)
(d) A mobile application with components supporting the acquisition and transmission of behavioural (e.g., recording of patient daily activities such as participation in conversations, watching TV), contextual (e.g., patient's location), cognitive (e.g., verbal reaction time) data as well as the notification and acceptance/rejection of decisions by the patient and/or their carers (decision selection component); and the execution n of periodic audiological and cognitive to collect the related data (audiological and cognitive test components).

IV. Specification of PHPDM Models

The specification of PHPDM models is ontology based. More specifically, PHPDM models are specified as instances of an ontology on an ontology that we have introduced for this purpose. The advocation of an ontology based approach for the specification of PHPDM models has been due to the ability of ontologies to provide an axiomatic foundation of the specification language and the realisation of corresponding reasoning processes that can aid decision making. The specific ontology that we have introduced for specifying PHPDM models has been based on three sub-ontologies, namely: (i) an ontology for governmental policy making (i.e., the G2G Ontology [13]); (ii) an ontology for data mining (i.e., the OntoDM-core ontology [14]); and (iii) an ontology for statistics (i.e., the STATO ontology [15]).

A view of the main classes of the ontology for specifying PHPDM models is presented in Figure 2. According to this ontology, public health policies are introduced to address or explore different ISSUEs. Examples of issues that can be the subject of a public health policy can be the introduction of a screening programme, the provision of certain types of medical devices to different types of patients or the introduction of a medical training programme for clinical professionals. In the policy example that we refer to in Section V, the issue of policy exploration is the provision of binaural or single HA to hearing loss patients with different characteristics.

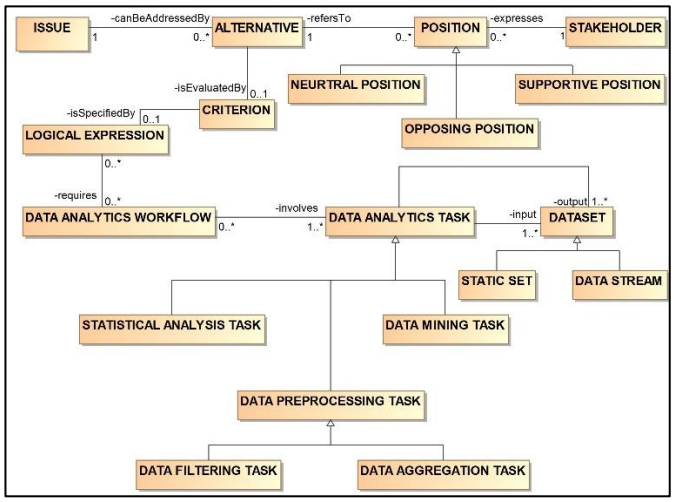


Figure 2 Main Classes of the Ontology Framework

Issues can be addressed by ALTERNATIVE decisions. An alternative (decision) presents a possible way of addressing theissue that is the subject of the policy. Alternatives reflect the key decisions that may be made in PHPDM process. They also need to be explored on the basis of evidence arising from the analysis of data. To express this, the ontology associates alternatives with a CRITERION that determines the circumstances under which the evidence arising from data analytics would support the alternative. A CRITERION is specified by a LOGICAL EXPRESSION over the outcomes of DATA ANALYTICS WORKFLOWS. Each DATA ANALYTICS WORKFLOW involves one or more DATA ANALYTICS TASKs which determine the way in which the available data are to be analysed in order to produce the evidence required by the criterion of the alternative.

A DATA ANALYTICS TASK takes as input one or many DATASETs and has output one or more other DATASETs. A DATASET can be STATIC data SET (i.e., a set that does not change frequently) or a DATA STREAM (i.e., a data set that changes continually). With regards to the processing that it performs upon its input data set(s), data analytics tasks can be distinguished into STATISTICAL ANALYSIS TASKs (i.e., tasks that carry out some statistical analysis upon the data), DATA MINING TASKs (i.e., tasks that carry out some data mining analysis upon the data), or a DATA PREPROCESSING TASK. The latter tasks are further distinguished into DATA FILTERING TASKs and DATA AGGREGATION TASK, i.e., tasks that filter data or aggregate data, respectively. Statistical analysis tasks are modelled using the STATO ontology and data mining tasks are specified using the Onto-DM ontology.

Initially, a PHPDM model includes a specification of tasks and the input data sets that they will be applied to. When a PHPDM model is executed, the description of it is expanded by including the data sets, which are produced by the data analytics workflow, as we explain in Section V.

The alternatives specified in a PHPDM model are related to POSITIONs taken by the different participants of the process (i.e., the STAKEHOLDERs). A position expressed by a stakeholder can be a SUPPORTIVE position (i.e., a position that supports the advocation of the alternative), an OPPOSING position (i.e., a position that is negative to the advocation of the alternative) or a NEUTRAL position (i.e., a decision indicating that the stakeholder neither supports nor objects to the alternative). A stakeholder may express SUPPORTIVE, OPPOSING or NEUTRAL positions for one or more alternatives but cannot express two different POSITIONS for the same alternative.

As discussed above, we use the OntoDM-core ontology [14], to specify in a formal manner the key data mining processes that should be applied to data in order to produce evidence. For this purpose, we have extended the OntoDM-core ontology by introducing classes to representing the different data mining algorithms of WEKA [16]. Each algorithm is represented by a distinct class, which has the appropriate properties for the configuration of the WEKA execution required for the algorithm.

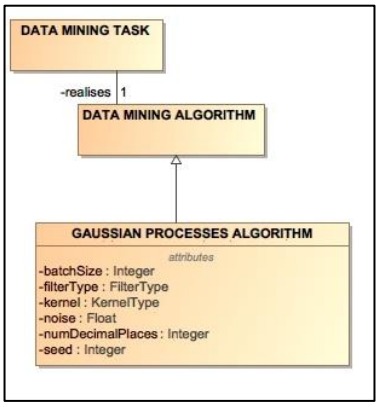


Figure 3 Example of Data Mining Algorithm Specification

An example of such classes, which has been introduced to represent the Gaussian Processes algorithm is shown in Figure 3. This class represents the properties that may be used to determine how the Gaussian Processes algorithm is to be applied, namely:

(i) the *batchSize* – i.e., the preferred number of instances to process if batch prediction is being performed
(ii) the *filterType* – i.e., whether the data should be transformed (the available options/values for this are: Normalize training data, Standardize training data, No normalization/standardization).
(iii) the *kernel* – i.e., the similarity function over pairs of data points in raw representation to use (available options in WEKA are: NormalizedPolyKernel, PolyKernel, PrecomputedKernelMatrixKernel, Puk, PBFKernel, StringKernel)
(iv) the *noise* – the level of Gaussian Noise (added to the diagonal of the Covariance Matrix, after the target has been normalized/standardized/left unchanged)
(v) the *numDecimalPlaces* – i.e., the number of decimal places to be used for the output of numbers in the model
(vi) the *seed* – i.e., the random number seed to be used

For the development of our ontology framework we merged the above three sub-ontologies using Protégé [17], based on the ontology merging approach described [18].

V. Example and Implementation

In this section, we present an example of specifying a PHPDM model using the ontology introduced in Section IV to realise a BDA process and explore alternative PHP issues.

Our example is related to policy regarding the provision of HAs and has been inspired by a policy statement of the Hearing Loss Association of America (HLAA) regarding the provision of binaural or monaural HAs[19] (HLAA recommends the provision of binaural HAs). Such a PHP decision could be based on evidence about the potential benefits from the provision of binaural or monaural HAs and, in particular, whether one of these two alternatives would affect the average daily usage of HAs by different types of patients with hearing loss. To enable the assessment of these alternatives, a BDA process can be set up to establish whether the average daily usage of HAs depends on characteristics of users including their gender, age and type of hearing loss. Exploring this issue could be based on a process, which as shown in Figure 4, involves the following steps:

(i) The specification of PHPDM model as an instance of the ontology introduced in Section IV to identify: the (policy) issue, the alternatives for addressing it, the data and the data analytics tasks that will be used to produce evidence from these data in order to explore each alternative, and the criteria for selecting amongst the different alternatives.
(ii) The execution of the data analytics tasks of the model specified in (1) and the recording of its outcomes as instances of the PHPDM model.
(iii) The querying the PHPDM model to identify alternatives with satisfied criteria

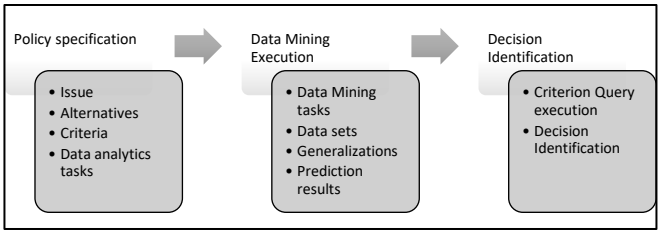


Figure 4 Implementation Overview

In the PHPDM model used in the above process:

- The Issue is Binaural Fitting.
- The possible Alternatives are: (ALT 1) binaural HAs to become the normal fitting practice, (ALT 2) no change in policies (i.e., use of monaural HA), and (ALT 3) other means of support to be considered (in case that none of alternatives (ALT 1) and (ALT 2) are clearly supported by the evidence generated by the BDA process.
- The Criteria are:
  - *Criterion 1*: Predicted average daily usage of single HA users to be less than predicted average daily usage of two HA users [(HA1+HA2)/2] with a minimum difference of 20% AND for binaural fitting users with issue delta: predicted average daily usage of single use period less than predicted average daily usage of two HAs use period AND NOT predicted average daily usage for all cases less than 2 hours (concerning ALT 1)
  - *Criterion 2*: Predicted average daily usage of single HA users to be greater than 120% of predicted average

daily usage of two HA users [(HA1+HA2)/2] AND for binaural fitting users with issue delta: predicted average daily usage of single use period greater than 120% of predicted average daily usage of two HA use period AND NOT predicted average daily usage for all cases less than 2 hours (concerning ALT 2)

- *Criterion 3*: Predicted average daily usage for all cases less than 2 hours (concerning ALT 3)

Furthermore, there are two data analytics tasks in the model: (1) to predict and compare the average daily usage of single HA users with a single HA vs those with two HAs (DAT_1), and (2) to predict and compare the average daily usage of binaural fitting users from the period of using one HA vs the period of using both HAs (DAT_2). Table I shows how we express the instances of the above-mentioned data analytics tasks in OWL.

TABLE I. INSTANCES IN OWL

```
Individual: DAT_1
      Types:
        data_analytics_task
Individual: DAT_2
      Types:
        data_analytics_task
```

Also, for the preparation of the mandatory data mining tasks the PHPDM model defines the following pre-processing tasks:

- a task to prepare dataset of single HA users
- a task to prepare dataset of two HA users
- a task to prepare dataset of the single period of usage of two HA users
- a task to prepare dataset of the two hearing-aid period of usage of two HA users

Additionally, we need a test set to get the predictions results from the execution of the generalisations. To achieve this, the PHPDM model includes also a task for preparing the test dataset with all the available combinations of the attributes.

The data mining tasks for the first data analytics task are: (i) to predict the average daily usage of single HA users and (ii) to predict the average daily usage of two HA users and for the second one are: i) to predict the average daily usage of the single period of usage of two HA users and ii) to predict the average daily usage of the two-HA period of usage of two HA users. For each of the above data mining tasks we also insert an instance of DM-dataset which contains the data produced by the pre-processing tasks described in the previous section.

We used the following regression algorithms of WEKA: (a) Gaussian Processes, (b) Linear Regression, (c) Multilayer Perception, (d) Support Vector Machine for Regression, (e) K-nearest neighbours, (f) K*, (g) Locally weighted learning, (h)Additive Regression, (i) Bagging, (j) Cross-Validation Parameter Selection, (k) Multischeme, (l) Random Committee, (m) Randomisable Filtered Classifier, (n) Random Subspace, (o) Regression by Discretisation, (p) Stacking, (q) Vote, (r) Weighted Instances Handler Wrapper, (s) Input Mapped Classifier, (t) Decision Table, (u) M5Rules, (v) Zero R, (w) Decision Stump, (x) M5P, (y) Random Forest, (z) Random Tree, and (aa) Rep Tree.

For each one of the 27 different algorithms, we added four different instances of data mining algorithm execution: one for each data mining task-dataset.

For each of the 108 (i.e., 4x27) algorithm executions, we added an instance of *Generalisation*, an object property of type *is_specified_output_of* to associate the instances with the data mining execution instances, and a data property with the path, where each model is stored. Along with the algorithm execution, we also performed 10-fold cross validation. For each instance of the Generalisation class, we also inserted a data property of type *has_output_details* which contains the textual output details produced by WEKA and an instance of the root mean squared error class, to store its value separately. To define which generalisation should be selected for execution for each data mining task, we need to query the ontology to get the algorithm executions with the minimum root mean squared error.

TABLE II. SPARQL QUERY

```
SELECT DISTINCT ?generalization ?dm_task WHERE {
?task rdf:type evotion:data_analytics_task.
?isabout rdfs:label "is about"@en.
?dm_task ?isabout ?task.
?algorithm_execution rdf:type  ontodm-core:OntoDM_000033.
?is_concretization_of rdfs:label "is_concretiza-
tion_of"@en.
?algorithm_execution ?is_concretization_of ?algorithm.
?is_specified_output_of rdfs:label "is_specified_out-
put_of"@en.
?generalization ?is_specified_output_of ?algorithm_execu-
tion.
?error ?is_about ?generalization.
?error ?has_value ?min_error_value{
    SELECT  (MIN( DISTINCT ?error_value) as ?min_er-
    ror_value) ?error WHERE {
    ?algorithm_execution rdf:type  ontodm-core:On-
    toDM_000033.
    ?is_concretization_of rdfs:label "is_concretiza-
    tion_of"@en.
    ?algorithm_execution ?is_concretization_of ?algo-
    rithm.
    ?is_specified_output_of rdfs:label "is_specified_out-
    put_of"@en.
    ?generalization ?is_specified_output_of ?algo-
    rithm_execution.
    ?is_about rdfs:label "is about"@en.
    ?error ?is_about ?generalization.
    ?has_value rdfs:label "has value"@en.
    ?error ?has_value ?error_value
    } GROUP BY ?error
}}
```

Table II shows the query (in SPARQL code) which returns, for each data mining task, the produced generalisation with the minimum root mean squared error. After executing the above query, we select which generalisations should be executed. The selected generalisations are executed for: (i) Single HA users, (ii) Two HA users, (iii) One HA period of Binaural fitting users, and (iv) Two HA period of Binaural fitting users. From the generalisation executions, the following result datasets are produced:

(i) result dataset of generalisation execution (Single HA users)
(ii) result dataset of generalisation execution (Two HA users)
(iii) result dataset of generalisation execution (One HA period of binaural fitting users)
(iv) result dataset of generalisation execution (Two HA period of binaural fitting users)

After executing the generalisations, we finally have the data needed to figure out whether the criteria defined are fulfilled.

For each *Criterion*, we proceed to a query execution and insert an instance of the Value class, with an object property containing a Boolean ("true" if the criterion is fulfilled, "false" otherwise). Finally, we proceed to the identification of the proposed policy decisions by extracting the alternative with Value: "true".

The data mining algorithm execution and the generalisation execution process is summarised in Figure 5.

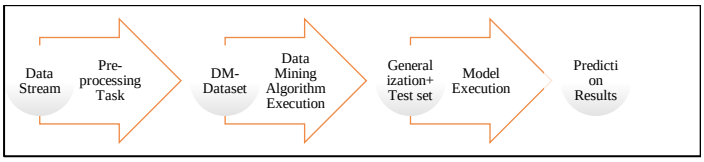


Figure 5 Implementation Tasks Execution

To test the above model, we used data from users of monaural HA and compared them to data from users with binaural HAs. The data that we used were synthetic. More specifically, we generated data for 1000 monaural HA users and 1000 binaural HA users. Also, we generated data from 1000 HA users, who initially had one HA and after some time got a second HA. In generating data, we assumed that the average daily usage data for each patient was from 3 months of the monaural aid use period and 3 months from the binaural HAs period. The generation of the synthetic data set did not consider the HA user acclimatisation period, as we considered data coming 8 weeks after issuing the HA/second HA. For all the constructed data, we generated the gender, the age, as well as the type of hearing loss randomly. To produce better data mining results and to validate our proof of concept we also made the following assumptions when generating the test data:

We generated the average daily usage according to the following conditions:

- if the type of hearing is low, the average daily usage is a random number integer between 0 and 19199 seconds,
- if the type of hearing is moderate, the average daily usage is a random number integer between 19200 and 38399 seconds, and
- if the type of hearing is severe, the average daily usage is a random number integer between 38400 and 57600 seconds.

Additionally, to manipulate the produced decision of our proof of concept, we assumed that for the single aid users the random numbers generated as described above were reduced by 30%. The same assumption was made also for the data produced for the single use period of the binaural fitting users.

Finally, we generated the test dataset, which includes all the combinations of the independent variables, to perform the generalisation executions and get the prediction results.

VI. Conclusions

In this paper, we have introduced an approach and a platform supporting public health policy decision making based on evidence produced by big data analytics processes. This approach is based on specifying public health policy decision making as instances of an ontology introduced for this purpose and executing them to support the public health policy decision making process. To demonstrate the use of our approach, we have used an example related to policy regarding provision of monaural or binaural HAs.

Our approach and the platform supporting it is under development. In particular, we are working on developing the scheme for transforming health policy decision making models into BDA processes onto specific BDA platforms (e.g. SPARK) and an experimental evaluation of our approach using real, non-synthetic data. We are also working on the full axiomatisation of the ontology for the specification of PHPDM models.

Acknowledgements

This work has been partly supported by the EU-funded project EVOTION (grant no H2020-727521).

References

[1] World Health Organization, "Key Policy Issues in Long-Term Care," World Heal. Organ., pp. 139–190, 2003.

[2] M. I. Wallhagen, "Access to care for hearing loss: Policies and stakeholders," J. Gerontol. Nurs., vol. 40, no. 3, pp. 15–19, 2014.

[3] J. Luo , et al., "Big Data Application in Biomedical Research and Health Care: A Literature Review.," Biomed. Inform. Insights, vol. 8, pp. 1–10, 2016.

[4] S. I. Hay, et al., "Big Data Opportunities for Global Infectious Disease Surveillance," PLoS Med., vol. 10, no. 4, 2013.

[5] J. V Tu, et.al., "The Cardiovascular Health in Ambulatory Care Research Team (CANHEART): using big data to measure and improve cardiovascular health and healthcare services," Circ Cardiovasc Qual Outcomes, vol. 8, no. 2, pp. 204–212, 2015.

[6] D. W. Bates, et al., "Big data in health care: Using analytics to identify and manage high-risk and high-cost patients," Health Aff., vol. 33, no. 7, pp. 1123–1131, 2014.

[7] T. B. Murdoch and A. S. Detsky, "The inevitable application of big data to health care," Jama, vol. 309, no. 13, pp. 1351–1352, 2013.

[8] J. Lavis, et al., "Towards systematic reviews that inform health care management and policy-making," J. Health Serv. Res. Policy, vol. 10, no. July, pp. 35–48, 2005.

[9] N. Mays, et al., "Systematically reviewing qualitative and quantitative evidence to inform management and policy-making in the health field," J. Health Serv. Res. Policy, vol. 10, no. suppl 1, pp. 6–20, 2005.

[10] L. Parker, "Including values in evidence-based policy making for breast screening: an empirically grounded tool to assist expert decision makers," Health Policy (New. York)., 2017, in press.

[11] A. Brennan, et al., "Developing policy analytics for public health strategy and decisions-the Sheffield alcohol policy model framework," Ann. Oper. Res., vol. 236, no. 1, pp. 149–176, 2016.

[12] S. Schneeweiss, "Learning from Big Health Care Data," N. Engl. J. Med., vol. 370, no. 23, pp. 2161–2163, 2014.

[13] E. N. Loukis, "An ontology for G2G collaboration in public policy making, implementation and evaluation," Artif. Intell. Law, vol. 15, no. 1, pp. 19–48, 2007.

[14] P. Panov, et al., Ontology of core data mining entities, vol. 28, no. 5–6. 2014.

[15] "STATO Ontology." [Online]. Available: http://stato-ontology.org.

[16] M. Hall, et al., "The WEKA data mining software," SIGKDD Explor. Newsl., vol. 11, no. 1, p. 10, 2009.

[17] N. F. Noy, et al., "Protégé-2000: An Open-Source Ontology-Development and Knowledge-Acquisition Environment," p. 2003, 2003.

[18] P. Lambrix, et al., "Evaluation of ontology development tools for bioinformatics," Bioinformatics, vol. 19, no. 12, pp. 1564–1571, 2003.

[19] HLAA, "Dispensing Hearing Aids." [Online]. Available: http://www.hearingloss.org/sites/default/files/docs/HLAA_POLICYSTATEMENT_Dispensing_Hearing_Aids.pdf.

Automatic Classification of Review Comments in Pull-based Development Model

Zhixing Li, Yue Yu* , Gang Yin, Tao Wang, Qiang Fan, Huaimin Wang
College of Computer, National University of Defense Technology, Changsha, 410073, China
{lizhixing15, yuyue, yingang, taowang2005, fanqiang09, hmwang}@nudt.edu.cn

***Abstract*—The pull-based model, widely used in distributed software development, allows any contributor to fork a public repository, package contributions as a pull-request, and then merge back to the original repository. Code review is one of the most significant stages in pull-based development. It ensures that only high-quality pull-requests are accepted, based on the in-depth discussion among reviewers. Thus, automatically identifying what reviewers are talking about in the discussions is benificial to better understand the code review process.**

In this paper, we conduct a case study on three popular open-source software projects hosted on GitHub and construct a fine-grained taxonomy including 11 sub-categories for review comments. We then manually label over 5,600 review comments, and propose a Two-Stage Hybrid Classification (TSHC) algorithm to classify review comments automatically by combining rule-based and machine-learning techniques. Comparative experiments with a text-based method achieve a reasonable improvement on each project (9.2% in Rails, 5.3% in Elasticsearch, and 7.2% in Angular.js respectively) in terms of the weighted average F-measure.

***Index Terms*—Pull-request; code review; review comment;**

I. Introduction

The pull-based development model is becoming increasingly popular in distributed collaboration for open-source software (OSS) development [4], [8], [10]. On GitHub [1] alone, the largest social-coding community, nearly half of collaborative projects (already over 1 million [9] in January 2016) have adopted this model. In pull-based development, any contributor can freely *fork* (*i.e.*, clone) an interesting public project and modify the forked repository locally (*e.g.*, fixing bugs and adding new features) without asking for the access to the central repository. When the changes are ready to merge back to the master branch, the contributors submit a *pull-request*, and then a rigorous code review process is performed before the pull-request get accepted.

Code review is a communication channel where integrators, who are core members of a project, can express their concern for the contribution [10], [20], [22]. If they doubt the quality of a submitted pull-request, integrators make comments that ask the contributors to provide several use cases or improve the implementation. With pull-requests becoming increasingly popular, most large OSS projects allow for crowd sourcing of pull-request reviews to a large number of external developers [23], [24] concerned about the development of the corresponding project to reduce the workload of integrators. After receiving the review comments, the contributor usually responds positively and updates the pull-request for another round of review. Thereafter, the responsible integrator makes a decision to accept the pull-request or reject it by taking all judgments and changes into consideration.

Previous research has shown that code review, as a well-established software quality practice, is one of the most significant stages in pull-based development [17], [18], [23]. It ensures that only high-quality codes are accepted. Several approaches have been proposed to study on how the code review process influences pull-request acceptance [19], [25] and latency [22], and software release quality [17]. However, a few studies have systematically investigated the automatic identification of what reviewers are talking about in the review discussions which is benificial to better understand the code review process.

In this paper, we conduct a case study on three popular OSS projects hosted on GitHub to comprehensively understand the categories of review comments from various stakeholders. First, we construct a fine-grained taxonomy covering the typical motivations of reviewers in joining the review process. Second, we manually label a large set of review comments according to the defined categories. With this dataset, we propose TSHC, a two-stage hybrid classification algorithm that is able to automatically classify review comments by combining rule-based and machine-learning (ML) techniques. The key contributions of this study include the following:

- A fine-grained and multi-level taxonomy for review comments in the pull-based development model is provided in relation to technical, management, and social aspects.
- A high-quality manually labeled dataset of review comments, which contains more than 5,600 items, can be accessed via a web page [2] and used in further studies.
- A high-performance automatic classification approach for review comments is proposed. The approach leads to a significant improvement in terms of the weighted average F-measure, namely 9.2% in Rails, 5.3% in Elasticsearch, and 7.2% in Angular.js, compared with the text-based method.

The rest of this paper is organized as follows. Section II introduces the pull-based development model. Section III elucidates the approach of our study. Section IV elaborates

* Corresponding author.
DOI: 10.18293/SEKE2017-039
[1]https://github.com/

[2]https://www.trustie.net/projects/2455

the research result. Section V presents related work, and Section VI provides the conclusion and future work.

II. Preliminary

On GitHub, a growing number of developers contribute to the open source projects by using the pull-request mechanism [8]. As illustrated in Figure1, a typical contribution process based on pull-based development model on GitHub involves the following steps.

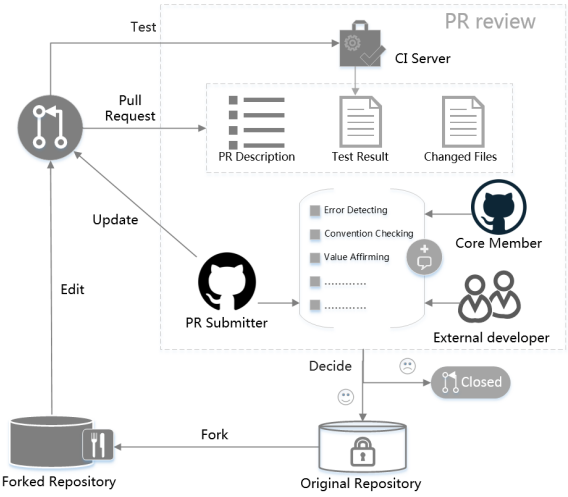


Fig. 1: Pull-based workflow on GitHub

Fork: Before contributing to an interesting project, the contributor has to fork the original project.

Edit: After forking, the contributor can edit locally without disturbing the main stream branch. He is free to do whatever he wants, such as implementing a new feature or fixing bugs according to the cloned repository.

Pull Request: When his work is finished, the contributor submits the changed codes from the forked repository to its source by a pull-request. Except for commits, the submitter needs to provide a title and description to explain the objective of his pull-request.

Test: Several reviewers play the role of testers to ensure that the pull-request does not break the current runnable state. They check the submitted changes by manually running the patches locally or through an automated manner with the help of continuous integration (CI) services.

Review: All developers in the community have the chance to review that pull-request in the issue tracker, with the existence of its description, changed files and codes, and test results. After receiving the feedback from reviewers, the contributor updates his pull-request by attaching new commits for another round review.

Decide: A responsible manager of the core team considers all the opinions of reviewers and merges or rejects the pull-request.

III. Approach

The goals of our work are to build a taxonomy for review comments in the pull-based development model and automate the comment classification according to the defined taxonomy.

A. Taxonomy Definition

Previous work has studied the challenges faced by pull-request reviewers and the issues introduced by pull-request submitters [10], [20]. Inspired by their work, we decide to comprehensively observe what reviewers are talking about in code reviews rather than merely focusing on technical and nontechnical perspectives.

We conducted a case study to determine the taxonomy scheme, which is developed manually through an iterative process of reading and analyzing review comments randomly collected from three projects hosted on GitHub. This process employs the following phases.

1) *Preparation phase*: The second author selects three projects (*i.e.*, Rails, Elasticsearch, and Angular.js) hosted on GitHub according to project popularity (*e.g.*, #star, #fork, and #contributor) and maturity (*e.g.*, project age, and pull-request usage). Table I shows the key statistics of the three projects in our dataset. Afterward, the first author randomly selects review comments from the three projects and labels each comment with a descriptive message.
2) *Execution phase*: The first author divides the previously labeled comments into different groups according to their descriptive messages. Unlike the aggregation process in card sorting [2], we initially set all comments to be in the same group and iteratively divide the group. Each comment is placed in the corresponding group(s). Through a rigorous analysis of existing literature and our own experience with working and analyzing the pull-based model in the last two years, we first identify the first level categories, which in turn are divided into other specific groups.
3) *Analysis phase*: By referring to the recognition result in the preceding phase, the second author abstracts taxonomy hierarchies and deduces general categories, which forms a two-level taxonomy scheme. The first and second authors, together with 10 other participants, verify the correctness and completeness of this scheme and make a final decision.

B. Two-Stage Hybrid Classification

To train the automatic classifier, we first manually labeled a set of review comments according to the defined taxonomy. For each project, we randomly sample 200 pull-requests (the comment count of which is greater than 0 and less than 30) per year from 2013 to 2015. Overall, 1,800 distinct pull-requests and 5,645 review comments are sampled. Based on this data set, we built TSHC which is illustrated by Figure 2. TSHC consists of two stages that utilize comments text and other information extracted from comments and pull-requests respectively.

TABLE I: Dataset of our experiments

Projects	Language	Application Area	Hosted_at	#Star	#Fork	#Contributor	#Pull-request	#Comment
Rails	Ruby	Web Framework	May. 20 2009	33906	13789	3194	14648	75102
Elasticsearch	Java	Search Server	Feb. 8 2010	20008	6871	753	6315	38930
Angular.js	JavaScript	Front-end Framework	Jan. 6 2010	54231	26930	1557	6376	33335

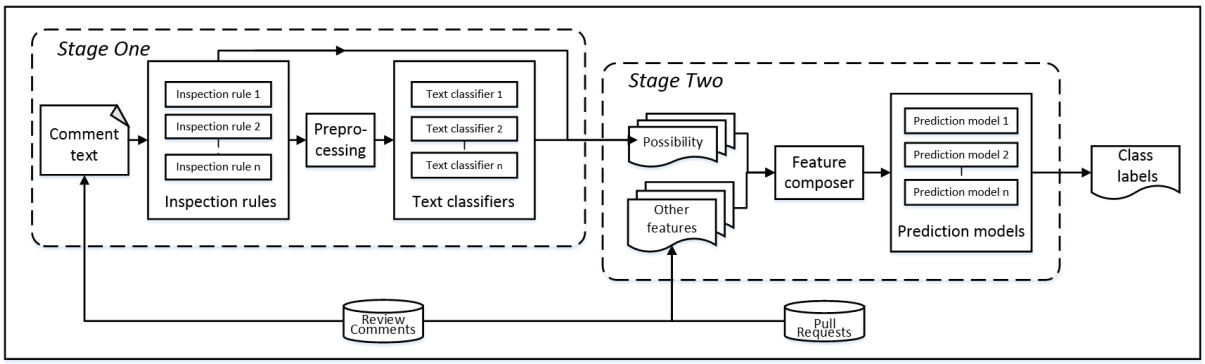


Fig. 2: Overview of TSHC

Stage One: The classification in this stage mainly utilizes the text part of each review comment and produces a vector of possibility (VP), in which each item is the possibility of whether a review comment will be labeled with the corresponding category.

Preprocessing is necessary before formal comment classification. Reviewers tend to reference the source code, hyperlink, or statement of others in a review comment to clearly express their opinion, prove their point, or reply to other people. These behaviors promote the review process but cause a great challenge to comment classification. Words in these reference texts contribute minimally to the classification and even introduce interference. Hence, we transform them into single-word indicators to reduce vocabulary interference and reserve reference information. Specifically, the source code, hyperlink, and statement of others are replaced with *'cmmcode'*, *'cmmlink'* and *'cmmtalk'* respectively.

After preprocessing, a review comment is classified by a rule-based technique, which uses inspection rules to match the comment text for each category. Several phrases often appear in the comments of a specific category. For instance, *"lgtm"* (abbreviation of *"looks good to me"*) is usually used by reviewers to express their satisfaction with a pull-request. This type of phrases are discriminating and helpful in recognizing the category of a review comment. Therefore, we establish an inspection rule for each category, which is a set of regular expressions abstracted from discriminating phrases. A category label is assigned to a review comment, and the corresponding item in VP is set to 1 if its inspection rule matches the comment text. The following shows examples of some regular expressions and the corresponding matched comments.

- Example 1
  - `(blank|extra) (line|space)s?`
  - *"Please add a new **blank line** after the include"*
- Example 2
  - `(looks|seem)s? (good|great|useful|awesome)`
  - *"**Looks good** to me, the grammar is definitely better."*
- Example 3
  - `(cc:?|wdyt|defer to|\br\?) (@\w+ *)+`
  - *"**/cc @fxn** can you take a look please?"*
- Example 4
  - `thanks?|thxs?|:(\w+)?heart:`
  - *"@xxx looks good. **Thank** you for your contribution **:yellow_heart:**"*

The review comment is then processed by the ML-based technique. ML-based classification is performed with scikit-learn[3], particularly the support vector machine (SVM) algorithm. The comment text is tokenized and stemmed to a root form [14]. We filter out punctuations from word tokens and reserve English stop words because we assume that common words play an important role in short texts, such as review comments. We adopt the TF-IDF (term frequencyinverse document frequency) model [3] to extract a set of features from the comment text and apply the ML algorithm to text classification. A single review comment often addresses multiple topics. Hence, one of the goals of TSHC is to perform multi-label classification. To this end, we construct text classifiers (TCs) for each category with a one-versus-all strategy. For a review comment which has been matched by inspection rule R_i (supposing that n categories of $C_1, C_2, \ldots, C_n$ exist), each TC ($TC_1, TC_2, \ldots, TC_n$), except for TC_i, is applied and predict the possibility of this review comment belonging to the corresponding category

Finally, the VP determined by inspection rules and text classifiers are passed on to *stage 2*.

Stage Two: The classification in this stage is performed on composed features. Review comments are usually short texts. Our statistics indicates that the minimum, maximum, and

[3]http://scikit-learn.org/

average numbers of words contained in a review comment are 1, 1527, and 32, respectively. The text information contained in a review comment is limited to be used to determine its category. Therefore, in addition to the VP generated in *stage one*, we also consider the following other features related to review comments.

`Comment_length`: This feature refers to the total number of characters contained in a comment text after preprocessing. Long comments are likely to argue about pull-request appropriateness and code correctness.

`Comment_type`: This binary feature indicates whether a review comment is inline comment or issue comment. An inline comment tends to talk about the solution detail, whereas an issue comment is likely to talk about other "high-level" issues, such as pull-request decision and project management.

`Core_team`: This binary feature refers to whether the comment author is a core member of a project or an external contributor. Core members are more likely to pay attention to pull-request appropriateness and project management

`Link_inclusion`: This binary feature identifies if a comment includes hyperlinks. Hyperlinks are usually used to provide evidence when someone insists on a point of view or to offer guidelines when someone wants to help other people.

`Ping_inclusion`: This binary feature refers to if a comment includes ping activity (occurring by the form of "@ username").

`Code_inclusion`: This binary feature denotes if a comment includes the source code. Comments related to the solution detail tend to contain source codes.

`Ref_inclusion`: This binary feature indicates if a comment includes a reference on the statement of others. Such a reference indicates a reply to someone, which probably reflects further suggestion to or disagreement with a person.

`Sim_pr_title`: This feature refers to the similarity between the text of the comment and the text of the pull-request title (measured by the number of common words divided by the number of union words).

`Sim_pr_desc`: This binary feature denotes the similarity between the text of the comment and the text of the pull request description (measured similarly as how `sim_pr_title` is computed). Comments with high similarity to the title or description of a pull-request are likely to discuss the solution detail or the value of the pull request.

Together with the VP passed from *stage 1*, these features are composed to form a new feature vector to be processed by prediction models. Similar to *stage 1*, *stage 2* provides binary prediction models for each category. In the prediction models, a new VP is generated to represent how likely a comment will fall into a specific category. After iterating the VP, a comment is labeled with class C_i if the i_{th} vector item is greater than 0.5. If all the items of the VP are less than 0.5, the class label corresponding to the largest possibility will be assigned to the comment. Finally, each comment processed by TSHC is marked with at least one class label.

IV. Result

A. Taxonomy

In the case study, we identified a fined-grained two-level taxonomy which is illustrated by Table II. There are four categories in the first level taxonomy, which are divided into more specific sub-categories respectively. The first category (*Correctness*) includes review comments which correct or imporve the quality of changed codes, while the second category (*Decision*) includes those that show reviewers' dicision of whether or not permitting the pull-request to merge back. In addition, reviewers are responsible to manage the review process (*Management*) and intertact with contributors (*Interaction*).

TABLE II: Complete taxonomy

Level-1	Level-2	Description
Correctness	**Style**	points out extra blank line, etc.
	Functionality	figures out functionality defect, etc.
	Test	demands submitter to provide test case, etc.
Decision	**Approval**	approves of the pull-request.
	Disagreeing	rejects to merge the pull-requestetc.
	Questioning	ask for more use cases .
Management	**Roadmap**	states the development roadmap , etc.
	Diversion	assigns other reviewers.
	Convention	ask for formulating commit messages, etc.
Interaction	**Response**	thanks for what other people do, etc.
	Encouragement	agrees with others' opinion, etc.

Although the first level taxonomy is more detailed than previous work, we refined it further in the second level taxonomy whose description can be seen in Table II. Moreover, Table III shows some of the example comments.

TABLE III: Example comments

Category	Example comments
Style	*"scissors: this blank line"*
Functionality	*"let's extract this into a constant. No need to initialize it on every call"*
Test	*"this PR will need a unit test, I'm afraid, before it can be merged"*
Approval	*"PR looks good to me. Can you ..."*
Disagreeing	*"I do not think this is a feature we'd like to accept."*
Questioning	*"Can you provide a use case for this change?"*
Roadmap	*"Closing as 3-2-stable is security fixes only now"*
Diversion	*"/cc @fxn can you take a look please?"*
Convention	*"Can you squash the two commits into one and also put [ci skip] in the commit message"*
Response	*"Thank you. This feature was already proposed and it was rejected."*
Encouragement	*":+1: nice one @cristianbica"*

B. Evaluation of TSHC

In the evaluation, we design a text-based classifier (TBC) as a comparison baseline. TBC uses the same preprocessing techniques and SVM models as used in TSHC. Classification performance is evaluated through a 10-fold cross validation, namely, splitting review comments into 10 sets, of which nine sets are used to train the classifiers and the remaining set is for the performance test. The process is repeated 10 times.

TABLE IV: Classification performance on Level-2 subcategories

Cat.	Rails						Elasticsearch						Angular.js					
	TBC			TSHC			TBC			TSHC			TBC			TSHC		
	Prec.	Rec.	F-M	Prec.	Rec.	F-M	Prec.	Rec.	F-M	Prec.	Rec.	F-M	Prec.	Rec.	F-M	Prec.	Rec.	F-M
Style	0.75	0.57	0.66	0.88	0.67	0.78	0.75	0.46	0.61	0.85	0.58	0.72	0.54	0.26	0.40	0.76	0.78	0.77
Functionality	0.63	0.84	0.74	0.71	0.86	0.79	0.67	0.92	0.80	0.77	0.82	0.80	0.65	0.71	0.68	0.69	0.71	0.70
Test	0.59	0.46	0.53	0.74	0.78	0.76	0.66	0.55	0.61	0.60	0.52	0.56	0.64	0.63	0.64	0.75	0.77	0.76
Approval	0.56	0.35	0.46	0.73	0.58	0.66	0.92	0.84	0.88	0.91	0.88	0.90	0.83	0.72	0.78	0.84	0.79	0.82
Disagreeing	0.50	0.26	0.38	0.63	0.43	0.53	0.65	0.36	0.51	0.80	0.67	0.74	0.63	0.48	0.56	0.61	0.51	0.56
Questioning	0.47	0.21	0.34	0.60	0.36	0.48	0.33	0.11	0.22	0.38	0.26	0.32	0.45	0.51	0.48	0.47	0.49	0.48
Roadmap	0.79	0.66	0.73	0.79	0.72	0.76	0.75	0.39	0.57	0.75	0.68	0.72	0.49	0.31	0.40	0.83	0.64	0.74
Diversion	0.83	0.77	0.80	0.98	0.78	0.88	0.57	0.35	0.46	0.88	0.90	0.89	0.35	0.16	0.26	0.96	0.67	0.82
Convention	0.83	0.76	0.80	0.90	0.81	0.86	0.94	0.57	0.76	0.96	0.76	0.86	0.85	0.76	0.81	0.83	0.82	0.83
Response	0.98	0.93	0.96	0.99	0.98	0.99	0.91	0.84	0.88	0.93	0.95	0.94	0.90	0.80	0.85	0.94	0.93	0.94
Encouragement	0.80	0.66	0.73	0.89	0.87	0.88	0.87	0.67	0.77	0.92	0.91	0.92	0.93	0.62	0.78	0.98	0.92	0.95
AVG	0.71	0.67	**0.69**	0.80	0.76	**0.78**	0.79	0.75	**0.77**	0.83	0.81	**0.82**	0.71	0.64	**0.67**	0.76	0.73	**0.75**

Table IV shows the precision, recall, and F-measure provided by different approaches for Level-2 categories. Our approach achieves the highest precision, recall, and F-measure scores in all categories with only a few exceptions.

To provide an overall performance evaluation, we use the weighted average value of F-measure [26] of all categories by the proportions of instances in that category. Equation 1 describes the formula to derive the average F-measure. In the equation, the average F-measure is denoted as F_{avg}, the F-measure of the i_{th} category as f_i, and the number of instances of the i_{th} category as n_i.

$$F_{avg} = \frac{\sum_{i=1}^{11} n_i * f_i}{\sum_{i=1}^{11} n_i} \quad (1)$$

The table indicates that our approach consistently outperforms the baseline across the three projects. Compared with that in the baseline, the improvement in TSHC running on each project in terms of the weighted average F-measure is 9.2% (from 0.688 to 0.780) in Rails, 5.3% (from 0.767 to 0.820) in Elasticsearch, and 7.2% (from 0.675 to 0.747) in Angular.js. These results indicate that our approach is highly applicable in practice.

We further study the review comments miscategorized by TSHC. An example is that *"While karma does globally install with a bunch of plugins, we do need the npm install because without that you dont get the karma-ng-scenario karma plugin."*. TSHC classifies it as belonging to *Functionality*, but it is actually a *Disagreeing* comment. The reason for this incorrect predication is twofold, namely, the lack of explicit discriminating terms and the too specific expression for rejection. Inspection rule of *Disagreeing* is unable to matched because of the lack of corresponding mathcing patterns. ML classifiers tend to categorize it into *Functionality* because the too specific expression of opinion makes it more like a low-level comment about code correctness instead of a high-level one about the pull-request decision.

We attempt to solve this problem by adding factors (*e.g.*, `Comment_type` and `Code_inclusion`) in *stage 2* of TSHC, which can help reveal whether a review comment is talking about the pull-request as a whole or the solution detail. Although the additional information improves the classification performance to some extent, it is not sufficient to differentiate the two types of comments. We plan to address the issue by extending the manually labeled data set and introducing a sentiment analysis.

V. Related Work

A. Code Review

Code review is employed by many software projects to examine the change made by others in source codes, find potential defects, and ensure software quality before they are merged [5], [12]. Traditional code review, which is also well known as the code inspection proposed by Fagan [7], has been performed since the 1970s. However, its cumbersome and synchronous characteristics have hampered its universal application in practice [2]. With the occurrence and development of VCS and collaboration tools, Modern Code Review (MCR) [16] is adopted by many software companies and teams. Different from formal code inspections, MCR is a lightweight mechanism that is less time consuming and supported by various tools. While the main motivation for code review was believed to be finding defects to control software quality, recent research has revealed that defect elimination is not the sole motivation. Bacchelli *et al.* [2] reported additional expectations, including knowledge transfer, increased team awareness, and creation of alternative solutions to problems.

B. Pull-request

Although research on pull-requests is in its early stages, several relevant studies have been conducted. Gousios *et al.* [8], [15] conducted a statistical analysis of millions of pull-requests from GitHub and analyzed the popularity of pull-requests, the factors affecting the decision to merge or reject a pull-request, and the time to merge a pull-request. Tsay *et al.* [19] examined how social and technical information are used to evaluate pull-requests. Yu *et al.* [25] conducted a quantitative study on pull-request evaluation in the context of CI. Moreover, Yue *et al.* [24] proposed an approach that combines information retrieval and social network analysis to recommend potential reviewers. Veen *et al.* [21] presented PRioritizer, a prototype pull-request prioritization tool, which

works to recommend the top pull-requests the project owner should focus on.

C. Classification on Free Text

Several studies have been performed to analyze free text generated in the software development process. Antoniol *et al.* [1] conducted a survey on 1,800 issues from the BTS of three large open-source systems and concluded that the linguistic information contained in these issues is sufficient to distinguish "bug" issues from "non-bug" ones. Later on, Pingclasai *et al.* [13] used topic modeling to classify bug reports. Herzig *et al.* [11] conducted a fine-grained classification on 7,000 issue reports and analyzed the classification results of early research. Zhou *et al.* [26] proposed a hybrid approach by combining text-mining and data-mining techniques to automatically classify bug reports. Ciurumelea *et al.* [6] studied reviews on mobile apps, proposed an approach to automatically organize reviews according to predefined tasks (battery, performance, memory, privacy, etc.), and recommended the related source code that should be modified.

VI. Conclusion&Future Work

In this paper, we conduct a case study on three popular OSS projects hosted on GitHub, and construct a fine-grained taxonomy including 11 sub-categories for review comments. According to the defined taxonomy we manually label over 5,600 review comments and propose a two-stage hybrid classification algorithm to automatically classify review comments. The comparative experiment results show that our approach can return reasonably good results for most categories.

Nevertheless, TSHC performs poorly on a few Level-2 subcategories. More work could be done in the future to improve it. we plan to address the shortcomings of our approach by extending the manually labeled data set and introducing a sentiment analysis. Moreover, we will try to improve reviewer recommendation and pull-request prioritization based on the result in this paper.

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, 61432020, 61472430, 61532004).

References

[1] Giuliano Antoniol, Kamel Ayari, Massimiliano Di Penta, Foutse Khomh, and Yann Ga Neuc. Is it a bug or an enhancement?: a text-based approach to classify change requests. In *Conference of the Centre for Advanced Studies on Collaborative Research, October 27-30, 2008, Richmond Hill, Ontario, Canada*, page 23, 2008.

[2] A Bacchelli and C Bird. Expectations, outcomes, and challenges of modern code review. In *International Conference on Software Engineering*, pages 712–721, 2013.

[3] Ricardo A Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. China Machine Press, 2004.

[4] Earl T Barr, Christian Bird, Peter C Rigby, Abram Hindle, Daniel M German, and Premkumar Devanbu. Cohesive and isolated development with branches. In *International Conference on Fundamental Approaches To Software Engineering*, pages 316–331, 2012.

[5] Olga Baysal, Oleksii Kononenko, and Reid Holmes. Investigating technical and non-technical factors influencing modern code review. *Empirical Software Engineering*, 21(3):1–28, 2016.

[6] Adelina Ciurumelea, Andreas Schaufelbhl, Sebastiano Panichella, and Harald Gall. Analyzing reviews and code of mobile apps for better release planning. In *IEEE International Conference on Software Engineering*, 2016.

[7] Michael E Fagan. Design and code inspections to reduce errors in program development. In *Pioneers and Their Contributions to Software Engineering*, pages 301–334. Springer, 2001.

[8] Georgios Gousios, Martin Pinzger, and Arie Van Deursen. An exploratory study of the pull-based software development model. In *International Conference Software Engineering*, pages 345–355, 2014.

[9] Georgios Gousios, Margaret Anne Storey, and Alberto Bacchelli. Work practices and challenges in pull-based development: the contributor's perspective. In *International Conference Software Engineering*, pages 285–296, 2016.

[10] Georgios Gousios, Andy Zaidman, Margaret-Anne Storey, and Arie Van Deursen. Work practices and challenges in pull-based development: the integrator's perspective. In *Proceedings of the 37th International Conference on Software Engineering*, pages 358–368. IEEE, 2015.

[11] Kim Herzig, Sascha Just, and Andreas Zeller. *It's not a Bug, it's a Feature: How Misclassification Impacts Bug Prediction*. 2013.

[12] Shane Mcintosh, Yasutaka Kamei, and Bram Adams. An empirical study of the impact of modern code review practices on software quality. *Empirical Software Engineering*, 21(5):1–44, 2016.

[13] N. Pingclasai, H. Hata, and K. I. Matsumoto. Classifying bug reports to bugs and other requests using topic modeling. In *Asia-Pacific Software Engineering Conference*, pages 13 – 18, 2013.

[14] M. F Porter. *An algorithm for suffix stripping*. Morgan Kaufmann Publishers Inc., 1997.

[15] Peter C Rigby, Alberto Bacchelli, Georgios Gousios, and Murtuza Mukadam. *A Mixed Methods Approach to Mining Code Review Data: Examples and a study of multi-commit reviews and pull requests*. 2014.

[16] Peter C. Rigby and Margaret Anne Storey. Understanding broadcast based peer review on open source software projects. In *International Conference on Software Engineering*, pages 541–550, 2011.

[17] Patanamon Thongtanunam, Shane McIntosh, Ahmed E Hassan, and Hajimu Iida. Investigating code review practices in defective files: an empirical study of the qt system. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, pages 168–179. IEEE Press, 2015.

[18] Patanamon Thongtanunam, Shane Mcintosh, Ahmed E. Hassan, and Hajimu Iida. Review participation in modern code review. *Empirical Software Engineering*, pages 1–50, 2016.

[19] Jason Tsay, Laura Dabbish, and James Herbsleb. Influence of social and technical factors for evaluating contribution in github. In *ICSE*, pages 356–366, 2014.

[20] Jason Tsay, Laura Dabbish, and James Herbsleb. Let's talk about it: evaluating contributions through discussion in github. In *The ACM Sigsoft International Symposium*, pages 144–154, 2014.

[21] Erik Van Der Veen, Georgios Gousios, and Andy Zaidman. Automatically prioritizing pull requests. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, pages 357–361. IEEE Press, 2015.

[22] Yue Yu, Huaimin Wang, Vladimir Filkov, Premkumar Devanbu, and Bogdan Vasilescu. Wait for it: Determinants of pull request evaluation latency on github. In *MSR*, 2015.

[23] Yue Yu, Huaimin Wang, Gang Yin, and Charles X Ling. Reviewer recommender of pull-requests in github. In *2014 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 609–612. IEEE, 2014.

[24] Yue Yu, Huaimin Wang, Gang Yin, and Tao Wang. Reviewer recommendation for pull-requests in github: What can we learn from code review and bug assignment? *Information and Software Technology*, 74:204–218, 2016.

[25] Yue Yu, Gang Yin, Tao Wang, Cheng Yang, and Huaimin Wang. Determinants of pull-based development in the context of continuous integration. *Science China Information Sciences*, 59(8):1–14, 2016.

[26] Yu Zhou, Yanxiang Tong, Ruihang Gu, and Harald Gall. Combining text mining and data mining for bug report classification. *Journal of Software: Evolution and Process*, 2016.

Mining Developer Behavior Across GitHub and StackOverflow

Yunxiang Xiong†, Zhangyuan Meng†, Beijun Shen†, Wei Yin‡
†School of Software, Shanghai Jiao Tong University, Shanghai, China
‡China Aeronautical Radio Electronics Research Institute, Shanghai, China
Email: {bruinx, 602389789, bjshen}@sjtu.edu.cn, yin_wei@careri.com

***Abstract*—Nowadays, software developers are increasingly involved in GitHub and StackOverflow, creating a lot of valuable data in the two communities. Researchers mine the information in these software communities to understand developer behaviors, while previous work mainly focuses on mining data within a single community. In this paper, we propose a novel approach to mining developer behaviors across GitHub and StackOverflow. This approach links the accounts from two communities using a CART decision tree, leveraging the features from usernames, user behaviors and writing styles. Then, it explores cross-site developer behaviors through T-graph analysis, LDA-based topics clustering and cross-site tagging. We conducted several experiments to evaluate this approach. The results show that the precision and F-Score of our identity linkage method are higher than previous methods in software communities. Especially, we discovered that (1) active issue committers are also active question askers; (2) for most developers, the topics of their contents in GitHub are similar to that of their questions and answers in StackOverflow; (3) developers' concerns in StackOverflow shift over the time of their current participating projects in GitHub; (4) developers' concerns in GitHub are more relevant to their answers than questions and comments in StackOverflow.**

***Index Terms*—Identity Linkage; Developer Behavior Mining; Machine Learning; GitHub; StackOverflow**

I. Introduction

In recent years, software developers are intensively involved in open source software development communities (e.g. GitHub) and knowledge sharing communities (e.g. StackOverflow). As developers continuously contribute to or exchange ideas through these communities, a lot of development data and knowledge are accumulated there. According to the data from ghtorrent[1] and archive.org[2], there are more than 20 million repositories and 5 million developers in GitHub, 30 million posts and 6 million users in StackOverflow as of August 2016. It is a great opportunity to understand characteristics and working habits of software developers through analyzing and mining these data.

Previous studies mainly focus on mining data within a single community [1] [2]. Meanwhile, it is more beneficial to conduct cross-community behavior mining, as some deep behaviors and developer relations can be discovered. Thus researches have been conducted on linking accounts across software communities and then making analysis. For example, Vasilescu [3] tried to find some associations between software development and crowdsourced knowledge, although the cross-site analysis is still quite simple.

Generally, we are facing three challenges when mining developer behaviors across software communities:

1) Identity linkage problem. The traditional social network relies on user names and email addresses for linking identities between communities. However, StackOverflow has no longer provided users' email addresses. Thus it lacks strong evidence for linking users with the same identities between GitHub and StackOverflow.
2) Data heterogeneous problem. Data across different software communities is heterogeneous. For example, labels of repositories in GitHub are programming languages, but those of Q&As in StackOverflow are technical terms.
3) Association mining. After identity linkage, it is also challenging to find the associations of developer behaviors across GitHub and StackOverflow, and recover the latent, valuable information of these data.

To address these challenges, this paper proposes a novel approach to mining developer behaviors across GitHub and StackOverflow, as shown in Fig. 1. It consists of two phases: identity linkage and behavior mining. At the identity linkage phase, we extract the features from the developer profile and behavior data, including the similarity between usernames, user behaviors, and user writing styles. And then classification and regression tree (CART) algorithms are applied to link the accounts of developers between GitHub and StackOverflow. At the behavior mining phase, we raise three research questions for exploring the patterns on developer behaviors across these two software communities. Statistics, Natural Language Processing (NLP) and machine learning technologies are adapted to analyze and mine the merged developer behavior data.

Our main contributions are summarized as follows:

1) We propose an approach to mining cross-site developer behaviors. It links identities between GitHub and StackOverflow by leveraging features from usernames, user behaviors, and writing styles, using CART decision tree. And then it mines the merged developer behavior data to find some valuable observations.
2) We conducted several experiments to evaluate the mining approach. The results show that the precision and F-Score of our identity linkage method are higher than

DOI reference number: 10.18293/SEKE2017-062
[1]http://www.ghtorrent.org
[2]https://archive.org/download/stackexchange

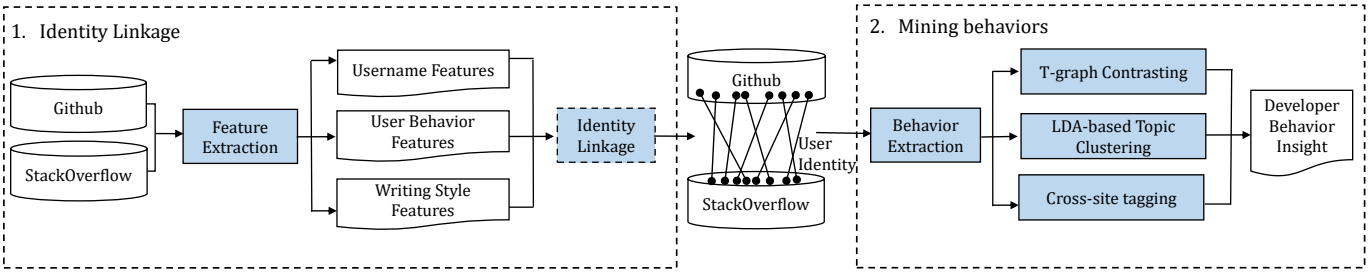


Fig. 1: Approach Overview

previous methods in software communities. Especially, we discovered that (1) active issue committers are also active question askers; (2) for most developers, the topics of their contents in GitHub are similar to that of their questions and answers in StackOverflow; (3) developers' concerns in StackOverflow shift over the time of their current participating projects in GitHub; (4) developers' concerns in GitHub are more relevant to their answers than questions and comments in StackOverflow.

II. Related Work

A. Identity Linkage in Software Communities

Several methods have been proposed to solve the identity linkage problem in software communities. A simple algorithm [4] was proposed by Goeminne using several simple rules to judge if two user pairs are the same person. Bird et al. [5] proposed a more advanced algorithm that some text similarity metrics are used, such as Levenshtein distance. Furthermore, a semantic-based method LSA was proposed to link users by Erik [6]. However, these methods didn't use username as the most important one of features to improve prediction accuracy, nor do they take full advantage of the textual information left by developers in software development.

B. Mining Developer Behaviors on Software Communities

There are many researches focusing on mining the data from a single community. For example, William [1] showed some developer behaviors and sentiments from open source repository mining; Christoph [2] argued that how do programmers ask ad answer questions on StackOverflow. On the other hand, Few people studied the association between these two communities. Bogdan [3] investigated the interplay between StackOverflow activities and the development process in GitHub. They show that the QA activity rate correlates with the code changing activity. So, across software communities, there are still a lot of patterns and insights to explore.

III. Identity Linkage across Software Communities

Accounts from GitHub and StackOverflow are less connected. Linking these accounts, which is called identity linkage, is a prerequisite of behavior mining across these two communities. In this section, we define the identity linkage problem, and then give our method, and its experimental results.

A. Problem Definition

Let P denote the collection of all natural persons, U_g denote the collection of users in GitHub and U_s denote the collection of users in StackOverflow. Let T(u) = p be a mapping function to map a user in GitHub or StackOverflow to a nature people in the real world. Now our goal is to find such a function f to solve the identity problem. For a user pair(c_1,c_2), where $c_1 \in U_g$ and $c_2 \in U_s$, $f(c_1, c_2) \rightarrow \{0, 1\}$. If T($c_1$) = T($c_2$), which means c_1 and c_2 refer to the same nature people, $f(c_1, c_2)$ equals 1, else $f(c_1, c_2)$ equals 0.

B. Feature Extraction

To solve the identity linkage problem defined above, we extract three kinds of features to calculate the similarities between two users firstly. Then we use a ground-truth data collection based on all these features as the input data and apply CART Decision Tree, a machine learning algorithm, to train an identity linkage model. Finally, each user pair is assigned a probability by the model. Here, these features are presented in detail, which are username features, user behavior features, and writing style features.

1) Username Features: Since people usually use similar usernames in different communities, it is very significant to extract useful features from usernames. Four string matching algorithms are chosen to measure the username similarity features: (1) levenstein distance, which is the number of transfers needed to transfer one string to another one; (2) jaro winkler distance, which is commonly used for measuring the similarity of short strings especially for usernames; (3) longest common substring, which is the longest string which is a substring of two strings; (4) longest common subsequence, which is the longest subsequence common to two strings.

The value of the jaro winkler distance is in the range [0,1], and we compute the ratio and transfer the value of the other three methods in the same range. In the experiments, we will set all these metrics as features to predict the identity linkage using the decision tree model, and two of them with the best performance will be chosen.

2) User Behavior Features : Existing empirical studies about social behaviors (e.g., [7]) show that, a user's social behavior exhibits a surprisingly high level of consistency across different communities over a sufficiently long period of time. It is rational to hypothesize that two users in GitHub and StackOverflow correspond to the same nature people in real world if they have a high level of synchrony.

For each repository in GitHub or each question in StackOverflow, there is a tag labeled to describe the programming language or relational technologies. Therefore, we can obtain topics in user behaviors with these tags, and measure the similarity of two user behaviors by the distribution similarity of the topics in their behaviors.

However, the tagging systems in these communities are very different. For example, the number of tags in GitHub and StackOverflow are 57 and 21300 respectively. Tags in GitHub are marked by the system automatically but by users themselves in StackOverflow. To solve this problem, we converse those synonymous tags into the same tags based on a synonyms relation[3], and a common set of all tags both in GitHub and StackOverflow are extracted. Then these tags are used to build the distribution of topics in user behaviors.

Another problem we encountered is, people are not always using different communities at the same time so that amount of information could be missing in such a process. Therefore we propose a user behavior matching method inspired by bio-stimulation [8] to reduce the impact of that problem. The main idea of the bio-stimulation is that the maximum stimulation from a pooled signal set plays a significant role for perception. So we segment the user behaviors by different time period. For example, we divide developers' behaviors into four quarters for each year and three months for each quarter.

Suppose each user pair (u^1,u^2) where $u^1 \in$ GitHub and $u^2 \in$ StackOverflow. For each month, we obtain all the language tags of the projects u^1 in GitHub and get a tag distribution for the tags belong to the common set. At the same time, we also catch all the questions u^2 asked on the StackOverflow in this month and get a tag distribution. Then, a cosine similarity is used to measure the user behaviors similarity for the month, denoted as $s_{mr}(i)$. Following formula is defined to calculate the similarity of user behaviors for a quarter and for a year, where N is the number of months. In our experiment, we measure an average similarity of user behaviors per year, as the final behavior similarity between two users.

$$S_{mr} = \frac{1}{N}(\sum_{i=1}^{N}(s_{mr}(i))^q)^{\frac{1}{q}}, q \geq 1 \quad (1)$$

3) Writing Style Features: Most of user generated data in the GitHub and StackOverflow are textual. Some studies [9] [10] [11] have shown that the user's writing style can help to achieve reliable results in user recognition situations. So we apply the method in [11] to extract user writing style features, listed in table I. Then, the writing style similarity between two users is measured by KL-divergence, using those features.

C. Identity Linkage

With all the features above, we train a identity linkage model on a ground-truth data set, using classification and regression tree (CART). By this model, we can obtain the probability of whether the user pair refer to the same user. And then, the identity linkage problem is converted to a matching problem

[3]http://stackoverflow.com/tags/synonyms

TABLE I: WRITING STYLE FEATURES

Feature	Definition
Length	number of different words
Vocabulary richness	frequency of hapax legomena, ddis legomena
Word shape	frequency of words with different combinations of upper and lower case letters
Word length	frequency of words that have 1-20 characters
Letters	frequency of a to z, ignoring case
Digits	frequency of 0 to 9
Punctuation	frequency of . ? ! , ; " ( )
Function words	frequency of words like 'the', 'of', and 'then'

in bigraph. Suppose user u^1 in GitHub and each candidate user u^2 in StackOverflow, a conditional Heuristic Greedy Matching (HGM) is used to avoid false positive matching and finally generate a candidate user pair as (u^1,u^2).

For example, in Fig. 2, each user pair has a probability which is assigned by the CART decision tree. In HGM, as shown in Fig. 2(a), the information generation of each user is calculated firstly. Suppose user X is the owner of the most information and his best candidate is X'. User pair (X, X') will be the first linkage selected by HGM because (X, X') shares the highest probability. After selecting, X and X' will be deleted in the candidate list before our next matching. Finally, three user pairs (X, X'), (Y, Y') and (Z, Z') are matched shown in Fig. 2(b).

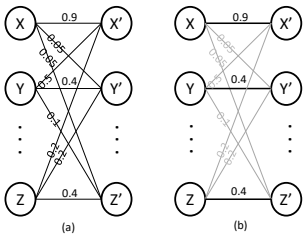


Fig. 2: An Example of Heuristic Greedy Matching Algorithm

D. Experiments

The experiment data is from ghtorrent[1] and archive.org[2] before May, 2016. To construct the ground-truth data, users' profile urls are used as the evidence to verify who are the same users. As the result, 16,000 users are linked corresponding to 8,000 people and we divide them into 5 groups and adopt 5-folder cross validation to evaluate the performance of our method.

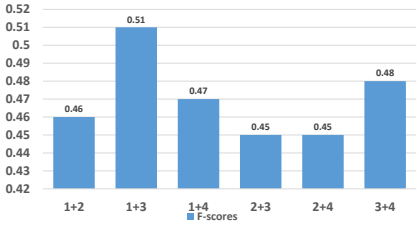


Fig. 3: Results for Combination of Different Username Metrics

1) Selection of Username Metrics: With the decision tree model, all potential combinations from 4 metrics are set as features to do prediction, and the best combination is selected. The result is illustrated in Fig. 3, where levenstein distance,

longest common substring, longest common subsequence and jaro winkler distance are abbreviated by 1,2,3 and 4 respectively. The experiment shows that the combination of levenstein distance and longest common subsequence has the best performance.

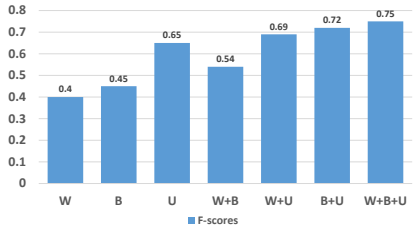


Fig. 4: Contribution for Different Features

2) Feature Contribution: In this experiment, all kinds of features above are used to train the decision tree model. They are username similarity (abbreviated by U), user behavior similarity (abbreviated by B), and user writing style similarity (abbreviated by W). Fig. 4 illustrates how the model is affected by each feature and the combination of features. It is demonstrated that the model trained with all features has the best performance.

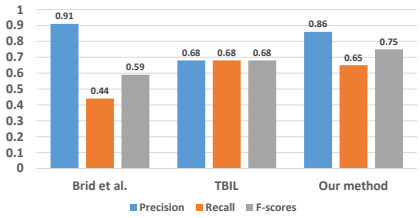


Fig. 5: Comparison with Other Methods

3) Comparison with Other Methods: We compare our method with the TBIL [12] method and the Bird's method [5]. Bird focuses on the username and email address of users. The TBIL is a state-of-the-art method, which uses usernames, user topics and user skills as features. Fig. 5 illustrates the result of the comparison. Bird's method achieves a high precision (around 0.91), but low recall and F-Score. That is because not all user use similar usernames among different communities and some users don't offer their email addresses. Since the TBIL uses a full matching strategy, it achieves a same precision, recall and F-Score (0.718). Our method has a good precision and recall, and get the best F-Score (around 0.75). Compared to those methods, our method adopts a bio-stimulation method with more features, which can analyze the user's topics and actions meticulously, and thus has a better performance. Furthermore, we use a conditional greedy-based matching to avoid false positive matching, since not all users have two different accounts in GitHub and StackOverflow.

IV. Mining Developer Behavior

After identity linkage, we will explore the developer behavior patterns across GitHub and StackOverflow in this section.

A. Research Questions

Three main research questions are designed as follows:

RQ1: Do one developer's activities in GitHub reflect his activities in StackOverflow? If yes, how?

We wonder if the developer's behaviors in GitHub and StackOverflow are relevant, and if the productivity of GitHub developer is relevant to his participation in StackOverflow. For example, are active issue committers in GitHub also active question askers in StackOverflow?

RQ2: How is the relevance of developers' concerned topics between GitHub and StackOverflow?

By our experience in software development, if a developer who participates in the development of a java project will encounter a lot of java errors or problems, then he is very likely to concern on java-related contents in knowledge-sharing community. Therefore, we attempt to know how is the relevance of developers' concerned topics between GitHub and StackOverflow and if the developers' concerns in StackOverflow will shift over the time of their current participating projects in GitHub. By collecting the developers' textual contents in GitHub and StackOverflow, we will measure the similarity between their topics.

RQ3: Which kind of activities in StackOverflow is more relevant to the developer's concerns in the software development process?

We will also explore which one will be closer to developers' concerned topics in GitHub among their questions, answers and comments in StackOverflow and which activity in StackOverflow is more representative of what developers concern in the software development process.

B. Behavior Mining

Guided by the above research questions, we collect and extract following developer behavior data from GitHub and StackOverflow: the number of developers' repositories in GitHub; the number of developers' questions, answers, comments and age in StackOverflow; descriptions (or readme file) of repositories, and issue data in GitHub; and textual contents of questions, answers and comments in StackOverflow. On this data, we conduct following analysis and mining:

1) T-graph Analysis: We summarize the results of T [13], a multiple contrast test procedure using 5% family-wise error rate, by means of T-graphs [14] showed in Fig. 7. The edges correspond to the results of the pairwise comparisons and nodes to the different groups being compared in such a directed acyclic graph. For example, if A have a higher value than B for a given metric, there is an edge from A to B (A→B).

2) LDA-based Topic Clustering: We merge all one's documents before applying LDA, because a user always generate more than one document. Firstly, Each document must be

converted to a bag of-words vector because the LDA method is a bag-of-words model. For each document, Stopwords[4] such as 'is' and 'the' need to be removed. Then, we use stemming to convert words into their root form[5] and remove some low frequency (no more than 20 times in all documents) words. After that, we set these text vectors of all documents as input, and apply the LDA method to get each user's topic distribution. We use the approach designed by [15] to decide the number of topics K by experiments. Finally, we measure the similarity of topic distributions by the symmetrical KL-divergence.

3) Cross-site Tagging: The tagging systems in StackOverflow and GitHub are very different. According to the principles set forth in [12], we can use a method to mark GitHub with tags in StackOverflow based on cross-site tagging which consists of two steps: (1) Unnecessary tags removal in StackOverflow. Interestingly, we found that 20% of tags could cover all questions in StackOverflow by experiments. Therefore, we remove some low frequency tags and rewrite the remaining 80% tags by some rules [12]. (2) Tag transfer from StackOverflow to GitHub. Since all questions and answers in StackOverflow (every answer is linked to a tagged question) are marked, these tagged data are used to train a naive Bayesian model for text classification. Then each repository in GitHub is labelled and gets a tag distribution TD using this model, based on the text contents in the readme file or project instruction.

After cross-site tagging, we calculate the co-occurrence of every two labels and apply the spreading activation to get more related labels. In one iteration (Fig. 6), the spreading activation method propagates corresponding similarity to other tags with its weight. And then, we measure the similarity of the tag distribution by symmetric KL-divergence.

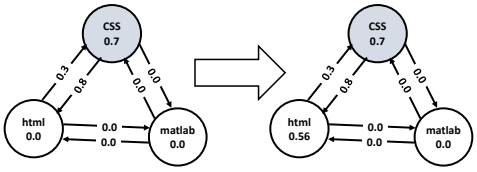


Fig. 6: An Example of Spreading Activation in One Iteration

C. Experiments and Findings

We conduct several experiments by the above technologies to answer three research questions.

For RQ1, we observe the distributions of the number of issues and questions. For each ordered pair of issues and questions, with the data sorted along one dimension, we split the other dimension into many groups and compare the distributions.

We perform experiments with our groups being quartiles. Fig. 8 shows that the most and second active 25% of the issue committers (Q1&Q2) ask more questions in StackOverflow than other quartiles (Q3&Q4), but Q1 and Q2 can not be distinguished. This phenomenon is consistent with the result using polynomial fitting as shown in Fig .9 that active issue committers are also active question askers.

[4]http://www.textfixer.com/resources/commonenglish-words.txt

[5]The stemming package is in http://www.nltk.org/api/nltk.stem.html

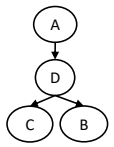


Fig. 7: T-graph

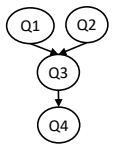


Fig. 8: Q1&Q2 Ask More Questions on SO than Others

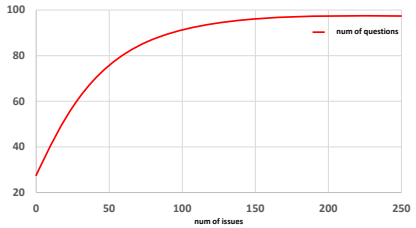


Fig. 9: Relationship Between the Num of Questions and the Num of Issues.

> Active issue committers are also active question askers.

For RQ2, we measure everyone's similarity of his textual contents between GitHub and StackOverflow by LDA and the tagging system. The similarities of developers are sorted from low to high, and illustrated by a fitting curve in Fig. 10. It's easy to observe in the figure that a small amount of developers' value is very low (only 0 to 0.28), but large number of developers have the similarity value from 0.3 to 0.5. The similarity between the contents of developers concerns in GitHub and StackOverflow is about 0.45. In addition, the experimental result shows that the cross-site tagging system outperforms the LDA method, because those tags maintained by StackOverflow already have a high degree of summary of the textual content.

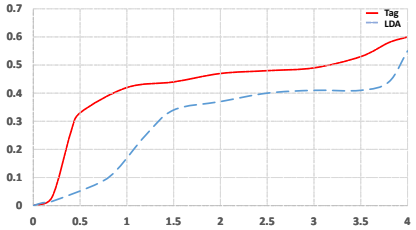


Fig. 10: The Similarity Measured by Cross-site Tagging System and LDA

> For most developers, the topics of their contents in GitHub are similar to that of their questions and answers in StackOverflow.

And then, we conduct in-depth research on RQ2. Suppose a developer participates in the project R, the readme file or description text of R is set to D_g, and the developer's direct participation in the project R starts at time T_s. Set the time interval Δt months, we obtain all questions, answers and comments of this developer in StackOverflow in $T_s \pm \Delta t$

and set to D_s. Then, the tagging system helps us to get the tag distribution for all D_s and D_g, and the KL-divergence is used to calculate the similarity between D_s and D_g. In this experiment, we set Δt to 4,6,8,10,12 and 14. And for each Δt, we simply filter out the upper and lower 10% extremum of the calculated similarities. The median value of the remaining similarity data is taken as the ordinate, and the abscissa is Δt as shown in Fig. 11. When Δt is 8 or 10, the contents that developers concerned about in GitHub and StackOverflow have the highest correlation, and the similarity is low when Δt is less than 8 or greater than 10. So, we speculate that developers will pay more attention to some of the project-related areas in a period of time.

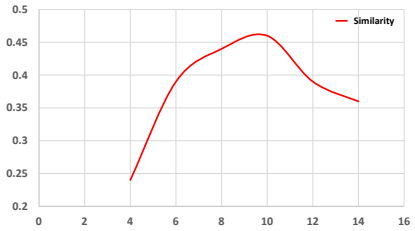


Fig. 11: The Similarity Changed With the Time of Projects

Developers' concerns in StackOverflow shift over the time of their current participating projects in GitHub.

For RQ3, we respectively compare the questions, answers, and comments of each developer in StackOverflow to the textual contents they left in GitHub. Fig. 12 shows that the textual contents that developers left in GitHub are more relevant to their answers than questions and comments in StackOverflow. In another word, the developer's answers are more representative of what developers concern in the software development process.

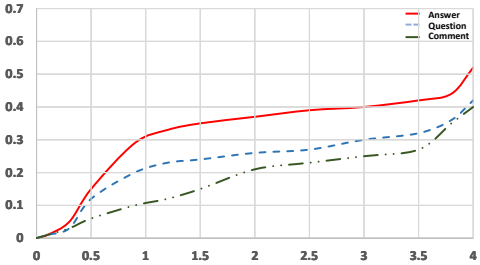


Fig. 12: Comparing Different Activities in SO to Contents in GH

The contents of developers' concerns in GitHub are more relevant to their answers than questions or comments in StackOverflow.

V. Conclusion and Future Work

In this paper, we propose a novel approach to developer behavior mining across GitHub and StackOverflow. It links the accounts from GitHub and StackOverflow, by leveraging the features from usernames, user behaviors and writing styles. Then, it mines developer behavior data across these two communities, and gains some valuable findings.

In the future, we plan to research the identity linkage problem among more than two software communities and continue to explore how StackOverflow influences GitHub, for example, what kind of issues or problem always be post to StackOverflow and what are the performance of software developers with different programming abilities in these two communities.

Acknowledgement

Beijun Shen is the corresponding author. This research is supported by National Natural Science Foundation of China (Grant No. 61472242) and 973 Program in China (Grant No. 2015CB352203).

References

[1] W. N. Robinson, T. Deng, and Z. Qi, "Developer behavior and sentiment from data mining open source repositories," in *49th Hawaii International Conference on System Sciences (HICSS)*, pp. 3729–3738, IEEE, 2016.

[2] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web?: Nier track," in *33rd International Conference on Software Engineering (ICSE)*, pp. 804–807, IEEE, 2011.

[3] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stackoverflow and github: Associations between software development and crowdsourced knowledge," in *International Conference on Social computing (SocialCom)*, pp. 188–195, IEEE, 2013.

[4] M. Goeminne and T. Mens, "A comparison of identity merge algorithms for software repositories," *Science of Computer Programming*, vol. 78, no. 8, pp. 971–986, 2013.

[5] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks," in *International Workshop on Mining Software Repositories (MSR)*, pp. 137–143, ACM, 2006.

[6] E. Kouters, B. Vasilescu, A. Serebrenik, and M. G. van den Brand, "Who's who in gnome: Using lsa to merge software repository identities," in *28th IEEE International Conference on Software Maintenance (ICSM)*, pp. 592–595, IEEE, 2012.

[7] G. Pickard, W. Pan, I. Rahwan, M. Cebrian, R. Crane, A. Madan, and A. Pentland, "Time-critical social mobilization," *Science*, vol. 334, no. 6055, pp. 509–512, 2011.

[8] S. Liu, S. Wang, F. Zhu, J. Zhang, and R. Krishnan, "Hydra: Large-scale social identity linkage via heterogeneous behavior modeling," in *ACM SIGMOD international conference on Management of data*, pp. 51–62, ACM, 2014.

[9] R. Zheng, J. Li, H. Chen, and Z. Huang, "A framework for authorship identification of online messages: Writing-style features and classification techniques," *Journal of the American society for information science and technology*, vol. 57, no. 3, pp. 378–393, 2006.

[10] A. Abbasi and H. Chen, "Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace," *ACM Transactions on Information Systems (TOIS)*, vol. 26, no. 2, p. 7, 2008.

[11] A. Narayanan, H. Paskov, N. Z. Gong, J. Bethencourt, E. Stefanov, E. C. R. Shin, and D. Song, "On the feasibility of internet-scale author identification," in *IEEE Symposium on Security and Privacy (SP)*, pp. 300–314, IEEE, 2012.

[12] W. Mo, B. Shen, Y. Chen, and J. Zhu, "Tbil: A tagging-based approach to identity linkage across software communities," in *Asia-Pacific Software Engineering Conference (APSEC)*, pp. 56–63, IEEE, 2015.

[13] F. Konietschke, L. A. Hothorn, E. Brunner, *et al.*, "Rank-based multiple test procedures and simultaneous confidence intervals," *Electronic Journal of Statistics*, vol. 6, pp. 738–759, 2012.

[14] B. Vasilescu, A. Serebrenik, M. Goeminne, and T. Mens, "On the variation and specialisation of workloadła case study of the gnome ecosystem community," *Empirical Software Engineering*, vol. 19, no. 4, pp. 955–1008, 2014.

[15] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *National Academy of Sciences*, vol. 101, no. suppl 1, pp. 5228–5235, 2004.

Exploring Identical Users on GitHub and Stack Overflow

Takahiro Komamizu, Yasuhiro Hayase, Toshiyuki Amagasa, Hiroyuki Kitagawa
University of Tsukuba, Japan
taka-coma@acm.org,{hayase,amagasa,kitagawa}@cs.tsukuba.ac.jp

Abstract

Analyzing behaviours of developers in different platforms (in particular, GitHub and Stack Overflow in this paper) can reveal interesting facts related to development activities. There are only few datasets for analysing cross-platform user behaviours, especially across GitHub and Stack Overflow. Users on GitHub and Stack Overflow are identifiable by equivalences of email addresses. In order to increase the number of identifiable users on these datasets, this paper retrieves potentially identifiable users between GitHub and Stack Overflow not relying only on email addresses. This paper employs a classification-based link prediction, which design the user identification problem as a link prediction problem on the bipartite graph consisting of users of GitHub and those of Stack Overflow. With the identification method, this paper generates a probabilistic dataset containing pairs of users with probabilities (or confidences). This paper, as well, publishes the identification tool in order to enable further data generation on appearing datasets of GitHub, Stack Overflow and others. The generated dataset and tool are highly helpful to accelerate researches on mining software repositories.

1. Introduction

Mining software repositories (or MSR) has become a largest and important research area not only in software engineering but also other data scientific research areas such as data mining and social network analysis. In particular, cross-platform analysis is a promising analytical task, however, few researches have studied is on software-relatred platforms. Software-related platforms including public software repository sites like GitHub[1] and Q&A sites like Stack Overflow[2] contain lots of knowledge related to software and programming. Many people are related to software repositories, for instance, software developers and software users, and they ask several questions on Q&A sites about software developments, software how-to, parameter settings of software, etc. Therefore, analysing not only software themselves but also activities of related people in other platforms may reveal highly important and useful facts for software engineering, for instance, users' activity relationship analysis on two or more platforms, and recommending related repositories and question-answers for improving developments in projects.

However, even though cross-platform analysis is emerging topic, available datasets related to software repositories are limited due to privacy issues and other concerns. In fact, there are lots of software repositories potentially on the Web, only a few of them however are open in public in the form of datasets (e.g., database snapshots of GitHub [1] and Stack Overflow [2]). Additionally, even if such databases are available, connecting two or more software repositories for cross-platform analysis is still problematic, especially, connecting two users on the different platforms is a hard task. This is mainly because of the lack of common identities of users. Names, affiliations, occupations, etc. do not solely work well to identify users on different repositories.

This paper attempts to connect users on GitHub [1] and Stack Overflow [2] whose datasets are published in the previous MSR conferences. A straightforward and sure matching approach to connect users on GitHub and Stack Overflow is that checking equivalences of MD5-hashed user emails of GitHub and pre-hashed user emails of Stack Overflow (as discussed in [3]). The number of users matched in the above matching approach is 53760, while that of GitHub users is 499,485 and that of Stack Overflow users is 1,295,620. This indicates that only 10.763% of GitHub users and 4.149% of Stack Overflow users are matched in the above matching approach. It is highly possible that more users can be matched, of whom cannot be matched via emails. Thus, this paper attempts to enhance the matching results with matchings which are not available only by email address information.

This paper reports a development of user identifications among repositories, namely, GitHub and Stack Overflow. This research models the user identification problem as a link prediction problem [4], which can also be modeled as a classification problem determining whether a pair of users is

[1] https://github.com/
[2] http://stackoverflow.com/

DOI reference number: 10.18293/SEKE2017-109

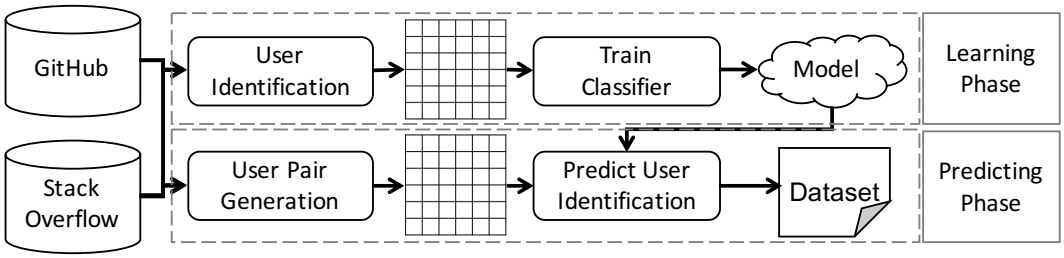


Figure 1: Overview of the proposed link prediction-based user identification. In the learning phase, user identification module processes users in GitHub and Stack Overflow datasets to identify users and generate a (user-pair $\times$ attribute-combination) matrix for learning which consists of similarities between users in terms of attributes. The predicting phase firstly generates a (user-pair $\times$ attribute-combination) matrix in the way analogous to the learning phase, and then trained classifiers in the learning phase predicts whether each pair of users is identical.

identical. The classification problem handles training data from [1] and [2], because users are identified by the same email addresses (this paper assumes that same users have same email addresses, and this assumption is same as [3]). The training data have only positive examples, so as to classify whether users are identical, the classification problem needs negative examples which users are surely not identical. Since there are far large number of negative examples comparing with that of positive examples, *data skewness problem* must be arisen. Data skewness problem is solved by down-sampling technique [5]. With the selected training data, this paper tries representative classification methods including linear regression [6], logistic regression [6], k-nearest neighbors [7], decision tree [8], and random forest [9], to observe which methods are feasible for user identification task. Experiments show that logistic regression, decision tree, and random forest are fairly better than other methods, and, interestingly, different weights of attributes are derives in the three methods (cf. Figure 2(a) and Figure 2(b)).

This paper mainly contributes to share the extracted datasets as well as a tool which identify users on GitHub and Stack Overflow using aforementioned identification methods. The datasets are probabilistic because of the result of statistical classification methods, however, the datasets are still useful for cross-platform analysis with handling probabilities. The user identification tool provides more possibilities to extend the datasets for published datasets of GitHub, Stack Overflow, and other software repositories. The classification methods can be replaced by appearing more sophisticated methods. The datasets and the tool are expected to accelerate research and developments related to software repository mining and other close fields.

Contributions of this paper are summarized as follows:

- **User identification mechanism** is proposed in this paper. The mechanism is inspired from classification-based link prediction methods. To realize successful user identification, this paper discusses the construction of training data by tackling with the data skewness problem.

- **Various classification methods are examined** in order to test which classification methods are appropriate to this task. The real dataset-based experimentation realizes that best classifiers are possible to identify users with 10% cross-validation errors.

- **Datasets and a tool** for user identificatiolns are made public and customizable. Therefore, the tools can be improved by public developers. For instance, classification modules can be replaced by more sophisticated modules, and extending identification methods by adding other datasets related to software engineering and others.

2. Link Prediction-based User Identification

This paper extends the datasets of GitHub [1] and Stack Overflow [2] by a supervised learning technique (i.e., a classification-based link prediction) to probabilistically identify users in each of them. This paper models user identification problem as a link prediction problem [4] which identifies presences of links between nodes in a graph. The user identification problem on users of GitHub and those of Stack Overflow is modeled as a link prediction problem on a bipartite graph. This work realizes the link prediction based on similarities between users.

Figure 1 overviews the proposed framework for link prediction-based user identification. The framework consists of two phases, learning phase and predicting phase. The learninig phase trains classifiers using obviously identifiable users who are identified by email addresses. The predicting phase identifies users using the trained models in the learinig phase. The following sections explain the detail of the framework including attribute selection, similarity computations via the attributes, and a prediction methodology using existing standard classification methods.

Table 1: Selected attributes on datasets. For GitHub dataset, attributes on `users` table and descriptions of `projects` table characterize GitHub users. For Stack Oveflow dataset, attributes of `users` table, question contents in `posts` table, and replies for questions in `comments` table.

Dataset	Table	Attribute	Type
GitHub	users	name	text
		location	text
		created_at	date & time
	projects	description	text
Stack Overflow	users	display_name	text
		location	text
		creation_date	date
		about_me	text
	posts	body	text
		tags	text
		title	text
	comments	comments	text

2.1. Attribute Selection on Each Dataset

The aim of this paper is to identify users on the datasets, thus user-related attributes are only necessary in the original complicated data. In GitHub data, this paper mainly uses `users` table which includes personal information registered on GitHub, and also includes project information because repository information indicate users' interests on developments. Similarly, in Stack Overflow data, this paper uses `users` table as well as questions and answers which also indicate users' interests on developments. Consequently, Table 1 for classification depicts selected attributes and their content types for each dataset. There are three types on the selected attributes, namely, `text`, `date & time`, and `date`.

In order to compute similarities between users, combinations of attributes are determined in a heuristic manner, and similarities for the combinations are calculated via well-studies metrics. This paper decides the combinations as shown in Table 2. Basic idea of the combinations are to combine semantically similar attributes. For instance, "users.name" in GitHub and "users.display_name" in Stack Overflow are combined because they both represent names of users, and "projects.description" in GitHub and "Stack Overflow.users.about_me" are describing interests of users, indeed, descriptions of projects related to users can represent users' interests.

2.2. Similarity Measures

Measuring similarities for the combinations denoted above are classified into (1) similarities between textual attributes, and (2) similarities between date & time and date. For textual similarities, various similarity functions are available (e.g., edit distance and cosine similarity between bags of words), and set-based similarity on trigram-based bag of words has achieved good matching performance. In the combinations (Table 2), this paper further divides the combinations of textual attributes into two, namely, "users.name" in GitHub and "users.display_name" in Stack Overflow, and other combinations. The similarity function for the former is cosine similarity (Equation 1) with trigram-based bag of words vectors, because names are relatively short and are same or similar in different services. That for the latter is cosine similarity (Equation 1) with TFIDF-based vectors.

Table 2: Combinations of attributes (b) for (user-pair $\times$ attribute-combination) matrix constructions. The combinations of attributes are selected based on similar contexts like `name` vs. `display_name`, `location` vs. `location`, `description` of projects vs. `body` of questions, etc.

Attributes on GitHub	Attributes on Stack Overflow
users.name	users.display_name
users.location	users.location
users.created_at	users.creation_date
projects.description	users.about_me
projects.description	posts.body
projects.description	posts.tags
projects.description	posts.title
projects.description	comments.comments

$$Cosine(\mathbf{v}_1, \mathbf{v}_2) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{|\mathbf{v}_1||\mathbf{v}_2|} \quad (1)$$

For the similarity between the date & time (i.e., "users.created_at" in GitHub) and date (i.e., "users.creation_date" in Stack Overflow), this paper defines a similarity function between dates by converting the date & time into date. The similarity function takes the inverse of the duration between dates, formally:

$$DateSim(date_1, date_2) = \frac{1}{|date_1 - date_2|} \quad (2)$$

2.3. Link Prediction

Given two users in GitHub and Stack Overflow, a classification-based link prediction classifies whether they will be connected in the future. There are a tremendous number of classification methods, and this paper selects fundamental classification methods (i.e., linear regression [6], logistic regression [6], k-nearest neighbors [7], decision tree [8], and random forest [9]), because they are well-known methods as well as they are available on popular machine learning libraries including scikit-learn[3],

[3] `http://scikit-learn.org/`

Spark[4], WEKA[5], and Mahout[6].

Classification is applied to a matrix which rows correspond with pairs of GitHub users and Stack Overflow users and columns correspond with combinations of attributes shown in Table 2. Elements in the matrix hold similarities of corresponding pairs of users on a combination of attributes.

For learning classifiers, training data are prepared from known identification method which is based on email addresses and their hashed values. As discussed in [3], users on GitHub and Stack Overflow are identified by the equivalences of MD5-hashed user emails of GitHub and pre-hashed user emails of Stack Overflow. The identified user pairs are used as positive examples. The negative examples are extracted based on identified users (denoted as I) by making pairs of identified users with other users. Formally, given a user $gu \in GitHub.users \cap I$ (or $su \in StackOverflow.users \cap I$), a negative example pair is (gu, nsu) where $nsu \in StackOverflow.users \backslash I$ (or (ngu, su) where $ngu \in GitHub.users \backslash I$).

2.4. Skewness Problem

Classification-based link prediction on the aforementioned training data easily fails, because of the data skewness problem. The data skewness problem is a famous problem on classification tasks, that is a classifier is trained to classify all labels as positive (or negative) when learning examples are skewed to positive (or negative) examples. This is because, in the training step of a classifier, it gets lower error rates if it classifies all examples into a major label.

In order to solve the skewness problem, this paper employs down-sampling technique [5]. Down-sampling are typical options to avoid the skewness problem, which equalizes the number of positive examples and that of positive examples. Suppose that the number of positive examples in the aforementioned dataset is M and that of negative examples is N where $M << N$ (in the fact of the dataset, $M = 53760$ and $N = 96.5$ billion), down-sampling performs random sampling on the negative examples in order to make $M \approx N$.

3. User Identification Quality Analysis

This section inspects the qualities of user identifications over the classification methods (i.e., linear regression, logistic regression, k-nearest neighbors, decision tree, and random forest) and discusses the balance of importance of the combinations of attributes to contribute to accurate classifications. The inspection is done with datasets of GitHub [1] and Stack Overflow [2]. The number of users in the former dataset is 499,485 and that of the latter dataset is 1,295,620. In the learning phase, 53760 users are used for positive examples and 107,520 users are used for negative examples (extracted as discussed in Section 2.4).

For determining best classification methods for this task, this paper examines the five classification methods by 10-fold cross-validation of the training examples. Figure 2(a) displays average cross-validation errors of classification methods. The figure indicates that random forest (RF), logistic regression (LG), and gradient boosting decision tree (GBDT) are better accuracy than linear regression (LR) and k-nearest neighbors (kNN), thus this paper employs RF, LG, and GBDT classifiers for the tool (Section 4)[7].

Interestingly, there are differences of learned weights for the combinations of attributes. Even though the selected classifiers achieve similar accuracy of classifications as shown in Figure 2(a), they have different preferences on the combinations. Figure 2(b) displays normalized weights (in order to make them comparable) indicating the importance of the combinations of attributes for accurate classifications. For instance, RF gives a higher weight on the combination of user names, which indicate that user names are the most important factor to identify users. For another example, GBDT indicates the combination of dates is the second most important factor. As the different weights are learning by these classifiers, they might provide different probabilistic datasets.

4. Dataset and Tool

The generated dataset and generating tool are available on project page PJD_GHSO[8] on GitHub. The dataset consists of a simple schema (namely, $\langle g\_user, s\_user, prob \rangle$), where g_user and s_user represent user IDs of GitHub and Stack Overflow, respectively, and $prob$ expresses the probability when these users are identical. For the first draft of published datasets, it contains 50k pairs of users classified by GBDT classifier.

The tool contains classifier learning modules and prediction modules. For the learning modules, they include similarity computation modules and learning modules. The former loads raw data to provide similarity measures for each combination of attributes on the pairs of training examples. The latter learns classifiers based on the similarities on the combinations of attributes and true labels of the pairs of users. The learned models are available in the tool and they are updatable for leaving space for improving classification accuracy. With the learned modules, prediction modules load them and predict labels for given pairs of users in GitHub and Stack Overflow. The prediction results are written in the aforementioned format.

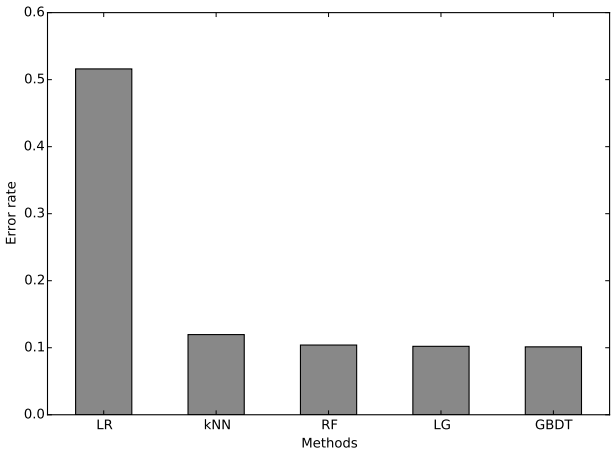
[4] http://spark.apache.org/

[5] http://www.cs.waikato.ac.nz/ml/weka/

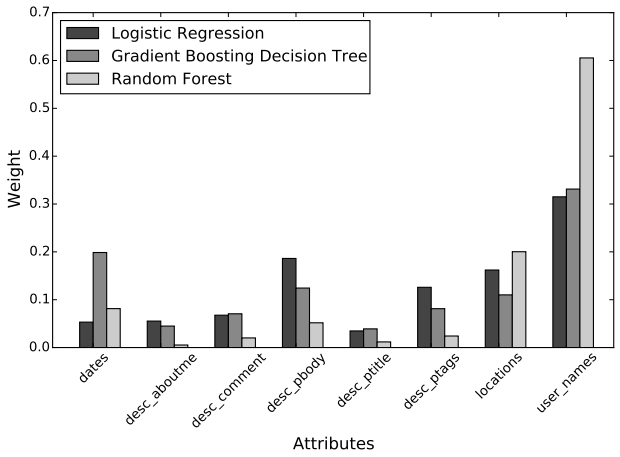
[6] https://mahout.apache.org/

[7] The published tool includes all of the classifier models.

[8] https://github.com/Taka-Coma/PJD_GHSO
Note that the project page is yet to be designed for public but for reviewing this paper. When this paper accepted, the page includes title of this paper, conference name, and other related information.



(a) Average cross-validation errors of classification methods. Comparing five classification methods: linear regression (LR), k-nearest neighbors (kNN), random forest (RF), logistic regression (LG), and gradient boosting decision tree (GBDT). LR is significantly worse than others and kNN are the second worst, and others are close and not significantly different. Consequently, RF, LG, and GBDT are better classifiers for the task.



(b) Normalized learned weights of attributes. The better classifiers in (a) are selected. The weights indicate the importance of each combination of attributes. Interestingly, the weights for each classification (LG, GBDT, and RF) are different. For example, RF considers much higher importance on similarities between user names for accurate classification, while GBDT considers that similarities between dates are important.

Figure 2: Experimental evaluations: (a) performance comparison of classification methods, and (b) learned weights for combinations of attributes for selected classification methods.

5. Discussion – Research & Application

The proposed tool and dataset increase research opportunities related to MSR, knowledge discovery, data mining, data management, etc. Developer behaviour analysis [3] is one of the fundamental research in order to reveal behavioural facts of developers on multiple platforms (i.e., GitHub and Stack Overflow). Vasilescu et al. [3] have analyzed associations of users' activity between GitHub and Stack Overflow. However, their work is limited to users identified by email address-based matching, thus the number of identified users are limited comparing with the proposed dataset in this paper. Therefore, they might miss some facts behind. The proposed dataset increases the opportunities of user behavioural analyses on GitHub and Stack Overflow in a probabilistic manner. Indeed, the dataset is not perfectly certain, but stochastic analyses can be used instead.

The dataset can be considered as an auxiliary dataset for applications on GitHub and Stack Overflow. For instance, repository recommendation on GitHub [10, 11, 12] has been studied to recommend relevant repositories to users. There are several ways of evaluating relevance of repositories (e.g., network analysis-based relevance, vector space similarities). For another example, user recommendation for review process on GitHub [13, 14] has been studied to smoothly managing developments. In general case of recommendation, lack of users' activities is critical for recommendation accuracy including the cold-start problem. In such situation, joining auxiliary datasets helps enhance users' activities on the dataset to the prior dataset, therefore, the joined dataset realizes better recommendation as well as solving the cold-start problem. Consequently, the proposed dataset acts as an auxiliary dataset to both GitHub and Stack Overflow datasets, and is expected to improve accuracy of applications.

6. Related Work

This paper aims at finding identical users on different platform, and this paper is, in the best of our knowledge, the first work of exploring identical users on GitHub and Stack Overflow and publishing datasets and identification tools. Wang et al. [15] have studied the user identification problem on different sites. Their approach only relies on user names by taking variations (like abbreviation) of user names into account for similarity computations between users on different platforms. They show that even only user names, moderate user identification accuracies can be achieved. While, this paper takes possible attributes into account to improve user identification accuracy, and the learned weights in Figure 2(b) indicate other factors are also important to identify users in different platforms. Motoyama et al. [16] gather attributes as sets of words and cal-

culate similarities among users based of the sets. They consider all attributes equally, however, this paper distinguishes attributes and classifiers take different importances for different combinations of attributes. Zhou et al. [17] have also studied user identification basically on social media network platforms like Twitter[9], Sina Microblog[10], Facebook[11], and RenRen[12]. They rely on topologies of social media networks, so their approach is not much applicable to datasets on GitHub and Stack Overflow. This is because users on GitHub (resp. Stack Overflow) are connected lesser than those on microblog-based social media networks. Zheng et al. [18] and Kong et al. [19] have been proposed content-based user identification. Zheng et al. [18] have identified users by their writing styles of messages on social media networks. The writing styles for individuals on GitHub and Stack Overflow are not surely evidential for identifying users between them, due to the differences of what to write as contents. While, Kong et al. [19] have attempted to identify users with spatio, temporal and textual similarity measurements with assumption of location-based social media networks. However, GitHub and Stack Overflow are not location-based services, therefore, their approach is not applicable.

7. Conclusion

This paper reports a user identification tool between GitHub users and Stack Overflow users by applying classification-based link prediction methods, and publishes a dataset of identified users with probabilities. This dataset expands the originally identified users on datasets of GitHub and Stack Overflow with email address-based identification. The proposed tool classifies with 10% errors in learning process.

This paper suggests two future works for improving the proposed tool in terms of quality and performance. There are several ideas for improving the quality: (1) ontological similarities on geographical attributes (i.e., locations) can provide more appropriate similarities on geographical attributes, and (2) more sophisticated classification and regression methods can improve the classification accuracy. The performance can be improved by two folds, scalability and storage. The learning module and predicting module should become faster by parallel and distributed computing (e.g., Hadoop, Spark or GPGPU). Because of large number of possible user combinations, large storage is necessary even for intermediate results.

[9] http://www.twitter.com
[10] http://www.weibo.com
[11] https://www.facebook.com/
[12] http://www.renren.com

Acknowledgement

This research was partly supported by the program *Research and Development on Real World Big Data Integration and Analysis* of RIKEN, Japan.

References

[1] G. Gousios, "The GHTorent Dataset and Tool Suite," in *MSR 2013*, 2013, pp. 233–236.

[2] A. Bacchelli, "Mining Challenge 2013: Stack Overflow," in *MSR 2013*, 2013.

[3] B. Vasilescu, V. Filkov, and A. Serebrenik, "StackOverflow and GitHub: Associations between Software Development and Crowdsourced Knowledge," in *SocialCom 2013*, 2013, pp. 188–195.

[4] D. Liben-Nowell and J. Kleinberg, "The Link-Prediction Problem for Social Networks," *Journal of the Association for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *JAIR*, vol. 16, pp. 321–357, 2002.

[6] D. A. Freedman, *Statistical Models: Theory and Practice*. Cambridge University Press, 2009.

[7] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," *IEEE Trans. Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[8] J. H. Friedman, "Stochastic Gradient Boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[9] T. K. Ho, "Random Decision Forests," in *ICDAR 1995*, 1995, pp. 278–282.

[10] M. Guendouz, A. Amine, and R. M. Hamou, "Recommending Relevant Open Source Projects on GitHub using a Collaborative-Filtering Technique," *IJOSSP*, vol. 6, no. 1, pp. 1–16, 2015.

[11] L. Zhang, Y. Zou, B. Xie, and Z. Zhu, "Recommending Relevant Projects via User Behaviour: An Exploratory Study on Github," in *CrowdSoft 2014*, 2014, pp. 25–30.

[12] T. Matek and S. T. Zebec, "GitHub open source project recommendation system," *CoRR*, vol. abs/1602.02594, 2016.

[13] M. M. Rahman, C. K. Roy, and J. A. Collins, "CORRECT: Code Reviewer Recommendation in GitHub Based on Cross-Project and Technology Experience," in *ICSE 2016*, 2016, pp. 222–231.

[14] 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 & Software Technology*, vol. 74, pp. 204–218, 2016.

[15] Y. Wang, T. Liu, Q. Tan, J. Shi, and L. Guo, "Identifying Users across Different Sites using Usernames," in *ICCS 2016*, 2016, pp. 376–385.

[16] M. A. Motoyama and G. Varghese, "I Seek You: Searching and Matching Individuals In Social Networks," in *WIDM 2009*, 2009, pp. 67–75.

[17] X. Zhou, X. Liang, H. Zhang, and Y. Ma, "Cross-Platform Identification of Anonymous Identical Users in Multiple Social Media Networks," *IEEE TKDE*, vol. 28, no. 2, pp. 411–424, 2016.

[18] R. Zheng, J. Li, H. Chen, and Z. Huang, "A framework for authorship identification of online messages: Writing-style features and classification techniques," *JASIST*, vol. 57, no. 3, pp. 378–393, 2006.

[19] X. Kong, J. Zhang, and P. S. Yu, "Inferring Anchor Links across Multiple Heterogeneous Social Networks," in *CIKM 2013*, 2013, pp. 179–188.

D3TraceView: A Traceability Visualization Tool

Gilberto A. de A. Cysneiros Filho
Department of Statistics and Informatics
Federal Rural University of Pernambuco
Recife, Brazil
g.cysneiros@gmail.com

Andrea Zisman
Centre for Research in Computing
The Open University
Milton Keynes, UK
andrea.zisman@open.ac.uk

***Abstract*— Software traceability is the ability to relate artefacts created during the life cycle of software development. Traceability is fundamental to support several activities of the software development process such as impact analysis, software maintenance and evolution, verification and validation. Despite the importance and advances in the software traceability area, traceability practice is still a challenge. One of these challenges is concerned with the visualization of traceability information. In this paper, we present D3TraceView, a traceability visualization tool that allows displaying traceability information in different formats depending on the purpose of use of traceability information. The tool supports different types of queries related to the use of traceability information. We use an Air Traffic Control Environment multi-agent system to demonstrate the use of the tool.**

Software Traceability; Information Visualization; D3.js

1. INTRODUCTION

Requirements traceability has been defined as the "*ability to describe and follow the life of a requirement, in both a forward and backward direction (i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases)*" [6][1]. This definition has been extended to incorporate the notion of software traceability [7] that encompasses and interrelates any uniquely identifiable software engineering artifact to any other[1].

The importance of traceability in the software development life-cycle has been endorsed by several standards for quality management and process improvement (CMMI, SPICE). Traceability is fundamental to support various activities of software development such as impact analysis, reuse, verification and validation; program comprehension; maintenance; and software evolution. Despite its importance, there are still several significant challenges associated with traceability, as described in [7]. In addition, in [3] the authors present several compelling areas of research that need to be addressed in order to support the challenges. One of these areas is concerned with the visualization of trace data. At the moment traceability information can be difficult to use since little attention has been given to the issue of presenting and visualizing traceability information.

As outlined in [12], the most common way to show traceability information in practice remains the use of matrix. Other ways to show traceability relations are tree browser explorer, lists, and hyperlinks views. As presented in the empirical study in [12], matrices and graphs are good to support management tasks; hyperlinks were preferred to support implementation and testing tasks. Matrices are adequate to provide an overview of the traces, and graphs could facilitate navigation through the traces. However, these forms of individually used visualizations have limitations to present traceability relations and assist with their use. The appropriate view of traceability information should be related to the task to be performed with the relations [22]. However, users are not always able to decide the forms of visualization that are most appropriate for the information they have, and for the tasks they need to use traceability information.

In this paper, we describe D3TraceView, a traceability visualization tool that provides different types of visualizations to users depending on the data to be visualized, the role of the user, and the task to which the user needs to perform with the relations. The tool is based on requirements identified on a previous work from one of the authors [5], and requirements defined in [16].

The main goal of the tool is to provide several types of visualization formats for specific trace data. The tool receives as input software artifacts (software models) and their respective traceability relations, and produces different views of the relations in various formats such as tables, matrices, radials, trees, sunbursts, etc. The tool also contains a repository of predefined queries related to the use of traceability relations. These queries are created to help users to execute several software activities, namely (i) software inspection (verification), (ii) maintenance and evolution (program comprehension), (iii) consistency checking of the models, and (iv) impact analysis. The tool was developed with Data-Driven Document (D3) JavaScript library (D3.js) [1].

The remaining of this paper is structured as follows. Section 2 provides a description of the D3TraceView tool. Section 3 explains the implementation aspects of the tool.

[1] Unless necessary to make a distinction, in this paper, we will use the term "traceability" to refer to both requirements and software traceability.

DOI reference number: 10.18293/SEKE2017-038

Section 4 discusses related work. Finally, Section 5 presents some conclusions and directions for future work.

2. D3TRACEVIEW TOOL

The goal of the D3TraceView is to be a visualization tool that provides interactive and context-specific traceability usage. In order to illustrate the tool and describe its main functionalities, we use examples of artefacts created during the development of a multi-agent system implementing an Air Traffic Control Environment (ATCE). The ATCE system was developed using *i** framework [23] to model the organizational environment; Prometheus methodology [19] to model system specification, architectural design, and detailed design phases; and JACK programming language [11] to implement the code. The traceability relations were generated using the rule-based traceability tool called RETRATOS [4], proposed by the authors of this paper.

D3TraceView was developed based on the challenge described in [13], in which traceability tools should be developed having user-friendly ways of formulating predefined and ad-hoc traceability queries, involving traces, artifacts, and all their relationships. Based on this concept, we developed D3TraceView to support a set of queries to help users to execute activities as proposed by OpenUP process [18]. Examples of these activities are: (i) assess results, (ii) software inspection, (iii) consistency checking, and (iv) change impact analysis.

The queries shown in this paper are concerned with artifacts of a multi-agent system, given the ATCE application we used to illustrate the tool. Examples of these queries are: (a) What are the i* actor elements that originate Prometheus goal elements? (b) What JACK agent elements are affected by changes in SD goal elements? However, other types of queries related to different activities, stakeholder's roles, and types of software artifacts could be specified and easily incorporated in the tool.

Apart from predefined queries, users can also analyze artifacts and trace relations browsing various visualization formats provided by the tool. This is possible by the use of mechanisms to zoom into some other data, as well as filtering retrieved data, and expanding and collapsing elements (e.g., matrix, table, radial, tree, sunburst).

A. Roles and Activities

Based on OpenUP process [18], we have defined several roles and activities to be supported by the tool. The OpenUP is an agile process for the development of software systems. We decided to base D3TraceView tool on OpenUP process due to: support for the whole system process development; being freely available; support for several roles that define the behavior and responsibilities of an individual, or a set of individuals in a development team; and (iv) support for development tasks.

The stakeholder's roles adopted by the tool are: (i) user/customer; (ii) business analyst; (iii) architect; (iv) designer; (v) project manager, and (vi) developer. The different activities currently implemented in the tool are: (a) assess results, (b) software inspection (verification), (c) maintenance and evolution (program comprehension), (d) consistency checking, and (e) change impact analysis. Table 1 shows a mapping of these activities and roles.

TABLE 1. MAPPING OF ACTIVITIES AND ROLES

Activities	Stakeholder's Role
Assess Results	Project Manager
Software Inspection	Business Analyst
Maintenance and Evolution	Architect, Designer, Developer
Consistency Checking	Architect, Designer, Developer
Change Impact Analysis	Business Analyst

B. Files, Queries and Visualization Formats

The screen in Figure 1 shows the main menu of options of D3TraceView. As shown in Figure 1, the user can (i) provide inputs about the software artifacts and traceability relation files to be used by the tool (option Files), (ii) choose pre-defined queries and visualize trace data in different formats that are associated with the queries (option Queries), and (iii) visualize traceability data in various different formats (option All Views). Figure 2 shows part of the screen to upload software artifacts to be used.

When a user chooses option "Queries" from the screen in Figure 1, the screen shown in Figure 3 is presented to the user displaying the activities supported by the tool. After selecting one of these activities the tool presents a set of pre-defined queries for the respective activity. Figures 4 to 8 show the screens with some of the queries for each of the various activities. For each activity, the user can select one or more queries. In addition, for some queries, the user can decide on a relevant artifact from a menu option. For example, as shown in Figure 6, in the case of the query "Who are the i* actors that originates Prometheus goals", the parts concerned with i* actors and Prometheus goals can vary depending on the artifacts being considered.

For each query, the tool presents its results as different graphs that are more appropriate to display the results of the query. However, it is possible to choose from a different type of visualization format (e.g., a visualization format to which the user is more familiar), by following the "All Views" option in Figure 1.

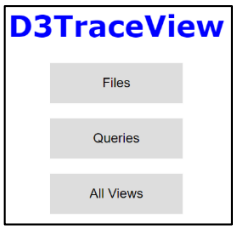


Figure 1. Main Menu

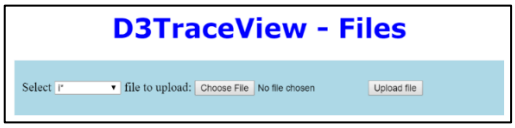


Figure 2. Upload of software artifact files

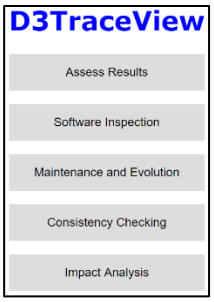


Figure 2. Menu of Activities

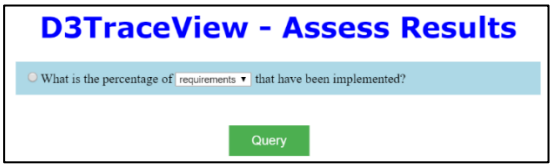


Figure 4. Assess Results

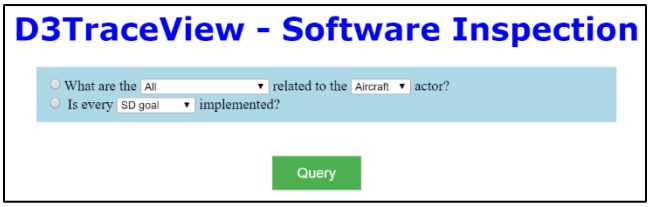


Figure 5. Software Inspection

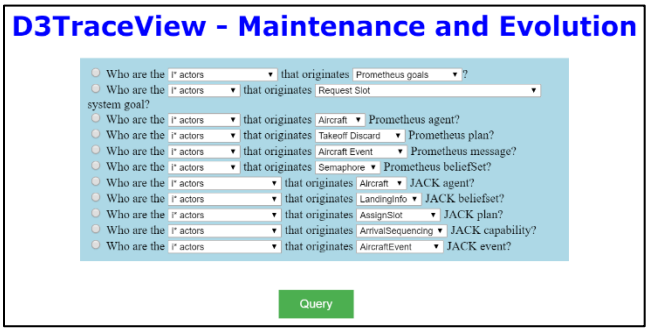


Figure 6. Maintenance and Evolution

Figure 7. Consistency Checking

Table 2 shows, for each pre-defined query, the types of visualization formats that the tool suggests. In Table 2, the identifiers of the queries follow the order in which the queries appear in the respective Figures 4 to 8. The types of visualization formats associated with the various queries were defined based on a comparative study presented in [12] and [17], and the available formats in D3 [1] that are adequate to present relations.

Figure 8. Impact Analysis

In the following we discuss the different types of queries associated with each activity and their respective visualization formats used to present the results of these queries.

TABLE 3. TYPES OF VISUALIZATION FORMATS BY QUERIES

Query	Type of Visualizations
Assess Results	
Q1	Bar graph, Gauge
Software Inspection (Verification)	
Q1	Table, Tree, Radial, Sunburst
Q2	Table, Tree, Radial, Sunburst, Matrix
Maintenance and Evolution (Program Comprehension)	
Q1 to Q11	Table, Tree, Radial, Sunburst, Matrix
Consistency Checking	
Q1	Table, Tree, Radial, Sunburst, Matrix
Impact Analysis	
Q1	Tree, Radial, Sunburst

Assess Results. In the OpenUP process [18], this activity is concerned with assessing the results of an iteration in the development process, and determining the success or failure of this iteration in order to plan a subsequent iteration. For this activity, the queries are concerned with the percentage of implemented artifacts. In this case, the tool presents the results as bar graphs or as gauge display.

Bar graphs are used to display and compare the number or frequency for discrete categories of data. Therefore, bar graphs are useful to present the percentage of certain element types. For example, in the case of a query concerned with the percentage of SD goals that has been implemented in the ATCE system, the SD goals, SD tasks, and system goals are the discrete categories of data. The results of this query can be shown as the number of total elements implemented with respect to the number of total elements, as shown in Figure 9.

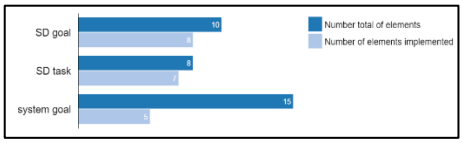


Figure 9. Bar graph view

Gauge displays are used to show a single discrete number representing the percentage. It is a progress bar with a circular, flat design. The gauge marker changes position within this range. Figure 10, shows the single discrete number representing the percentage in a gauge display type of graph.

Figure 10. Liquid fill gauge display

Software inspection (verification). Software verification is the collection of methods used to determine if a software system is correctly built. For this activity,

D3TraceView provides queries that compare artifacts of the system in terms of different relation types (e.g., implemented by, overlaps, achieves).

As shown in Figure 5, one of the queries is concerned with artifacts that are related to a certain type of element (in this case a certain type of actor). Another question is concerned with the case that verifies if a certain artifact is implemented in the system. The types of artifacts can vary depending on the documents (models) used for representing the system (e.g., i*, Prometheus, or JACK elements,); or other elements depending on the system. In these types of queries, the visualization technique shows traceability information involving specific artifacts.

As shown in Table 2, D3TraceView provides four visualization types for the first query namely table, tree, sunburst, and radial views. Tables are typical visualization formats to present simple lists of data (e.g. list of Prometheus goals). The disadvantage of using tables is that they do not show an overview of the dataset and the hierarchical information of traceability relations. Tables should also not be used to display large amount of data. Figure 11 shows a table with elements related to actor Aircraft.

Traceabilink Relationships

Source	Target
Find Best Landing Time for an Aircraft (SD Goal)	Landing (Goal)
Find Best Landing Time for an Aircraft (SD Goal)	Find Best Landing Time for an Aircraft (Goal)
Find Best Landing Time for an Aircraft (SR Goal)	Landing (Goal)
Find Best Landing Time for an Aircraft (SR Goal)	Find Best Landing Time for an Aircraft (Goal)
Landing (SR Task)	Landing (Goal)
Landing (SR Task)	Progresses an aircraft to Landing (Goal)
Landing (SR Task)	Query Best Landing Time from All Runway Manager (Goal)
Landing (SR Task)	Find Best Landing Time for an Aircraft (Goal)
Assign Slot (SR Task)	Assign Slot (Goal)
Initiate Approach (SR Task)	Initiate Aircraft Approach (Goal)
Process Schedule for a Feeder (SR Task)	Process Schedule for a Feeder (Goal)

Figure 11. Table view

Tree views are graphical representations to support hierarchical information display, as shown in Figure 12. In this case, each element can have a number of sub-elements. An element can be expanded to show sub-elements, if any exist, and collapsed to hide sub-elements. Tree views are very intuitive, largely used by software development tools, and allow elements to be shown without the need for scrolling the screen.

Sunburst is another type of graphical representation where nodes are drawn on adjacent rings representing a tree structure, as shown in Figure 13. In this representation, each child of a node with depth n is represented in the ring n + 1 on the same radian space as its parent(s). Sunburst performs better on large amounts of nodes than traditional tree view representations. Tree view representation grows rapidly in the vertical direction if many branches are expanded. In the case of sunburst, the nodes are distributed uniformly in all directions. In sunburst, the color of each item corresponds to an attribute of the item. In D3TraceView, the color in a sunburst view represents an element type (e.g. Prometheus goal), while the width of the representation of each element shows the importance of the element (i.e. number of sub-elements associated with the element). In addition, in the tool, sunburst view is dynamic, in which it is possible to click on an element and zoom information about this element, and it supports a large amount of data.

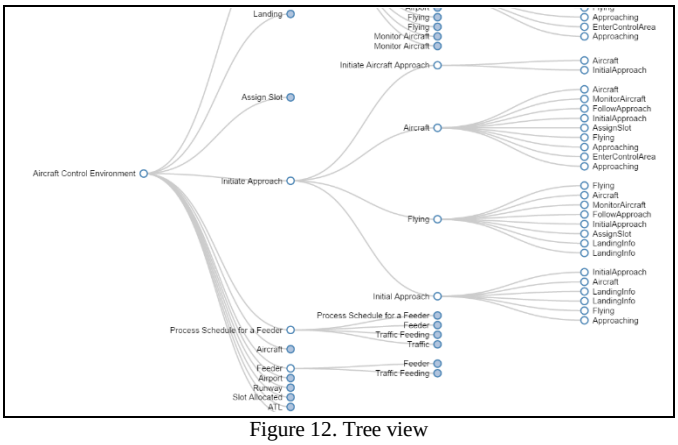


Figure 12. Tree view

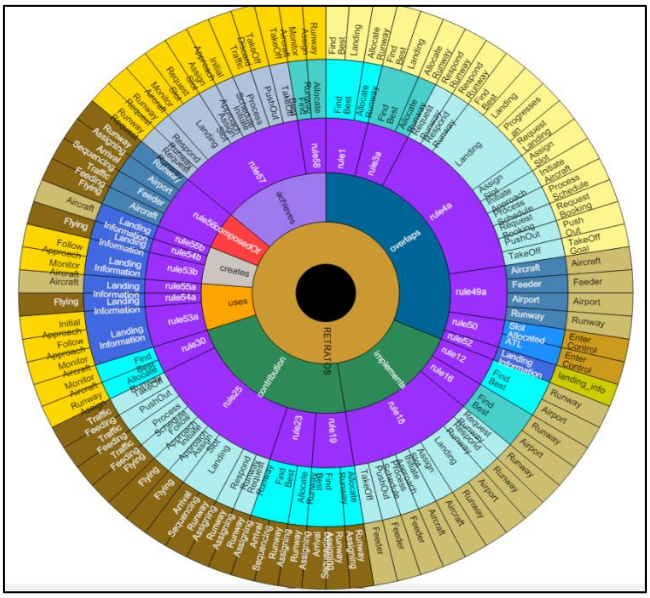


Figure 13. Sunburst view

Radial representation like Reingold-Tilford is a mixture of sunburst and tree views that do not use information of colors and spaces, as shown in Figure 14. For the second type of query, D3TraceView provides five visualization formats, namely list, matrix, sunburst, radial, and tree view. Figure 15 shows the matrix format, while all the other formats have been shown above.

Maintenance and evolution (Program comprehension). As shown in Figure 6, the pre-defined queries to support this activity are concerned with gathering a better understanding of the software. More specifically, the queries identify the artifacts in a certain software development phase that are related to artifacts in subsequent phases in the software development process. For example, i* actors (requirements phase) that are related to Prometheus goals (design phase). For these queries D3TraceView suggests five visualization types (see Table 2). The graphs for these visualization types are similar to the ones shown above.

Consistency Checking. Based on Figure 7, the queries to support this activity are concerned with the visualization of different types of relationships between various types of artifacts. Examples of these relationship types are: overlaps,

depends, contributes, uses, achieves, creates, and composes. Similar to the above activities and queries, the results of these queries can also be visualized in the five different visualization types shown above.

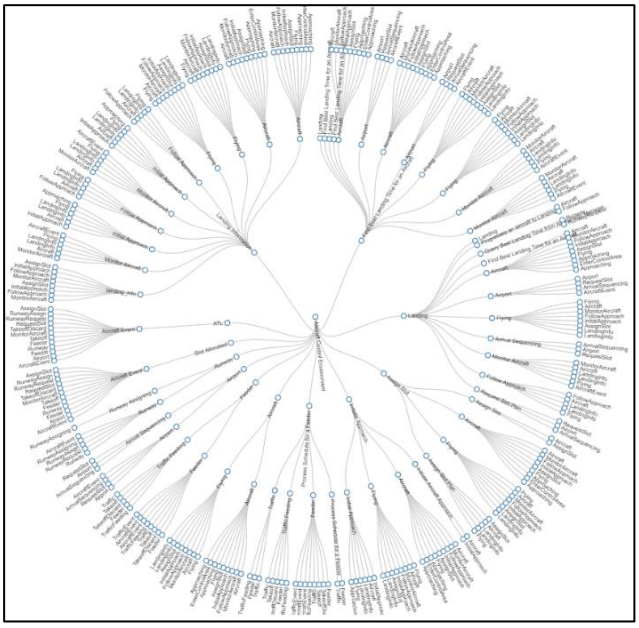


Figure 14. Reingold-Tilford tree view

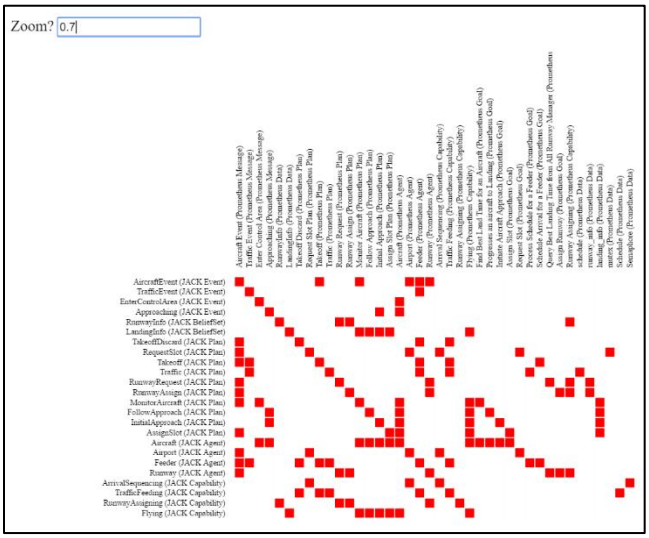


Figure 15. Matrix view

Impact Analysis. In this activity the queries are concerned with identifying the consequences of a change in the software, or identifying the artifacts that need to be changed. The tool suggests four visualization types, as described in Table 2.

3. IMPLEMENTATION

D3TraceView has been implemented as a web tool using D3.js library [1]. Figure 16 shows an overview of the main components of the tool. As shown in Figure 16, CASE tools represent the various tools that could be used to develop the software models. For our ATCE system, the case tools are Prometheus Design Tool (PDT), TAOM4E (for i* models), and JACK Intelligent Agent. The Traceability Generator represents tools that assist with generation of traceability relations (e.g., RETRATOS [4]). Both the software models and the traceability links are inputs to D3TraceView. The tool requires the links and the models to be represented in XML format, or in JSON format. The Generator component of D3TraceView generates the results associated with the various queries in the different visualization formats available in the D3 library.

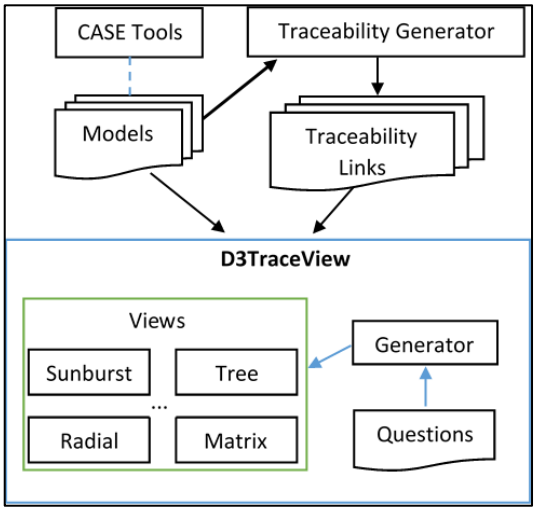


Figure 16. D3TraceView Component Diagram

4. RELATED WORK

Information visualization has been recognized as an important area of research that focuses on the use of visualization techniques to help people understand and analyze data [9]. It applies design principles, human perception, and color theory to present data. For several years, traceability information used to be visualized in terms of matrices and tables. However, a matrix is a two-dimensional view of a multidimensional information space (several software artifacts, with different levels of granularity, and various traceability relations connecting the artifacts). Moreover, when using a matrix it is difficult to navigate through the relationships. Another significant limitation is the lack of scalability [12]. In certain projects where the number of relationships is large, it becomes more difficult to visualize specific relationships. More recently, some other works have been developed to support information visualization of traceability information [16] [21] [10][20][2] [14] [15].

Marcus et al. [16] developed a prototype tool called TraceViz that is integrated with Eclipse IDE. TraceViz has three main areas to display: elements (source and target) in a hierarchical way; traceability relationships for a chosen element; and information of the properties and browsing history of a particular traceability relationship. It does not support traditional visualizations techniques such as matrix, tree, and radial. The technique in [21] is based on a graph-based representation with rings showing project artifacts and nodes representing relationships. The various types of relationships are indicated by the use of specific colors.

Similarly, Heim et al. [10] proposed an approach based on graphs representing requirements as nodes and relationships as vertices. However, the use of graph-based approaches to visualize traceability information are less intuitive to display relationships and do not scale well. In the case of large amounts of data it is difficult to understand the view as graphs, in which only a limited set of elements can be displayed. Merten et. al [17] use sunburst and netmap visualization techniques to show traceability relationships. These techniques were implemented as plugins for the Redmine project management tool. Thommazo et al. [20] present an approach to automated generation of requirement traceability matrix. In this work, traceability relationships can also be shown though a graph-based visualization. Chen et al. [2] present an approach that combines treemap and hierarchical tree visualization techniques to provide a global structure of traces and a detailed overview of each trace.

The above approaches are limited to support one or two types of visualization formats. D3TraceView supports several visualization formats and it has been developed in a way that it can easily accept other types of visualization formats. Moreover, our tool also allows for querying traceability data for different activities.

Several works have been suggested to support trace queries. A visual trace modeling language (VTML) was proposed in [14] that allows users to create queries using UML class diagrams and constraints. Maletic and Collard [15] propose a Trace Query Language (TQL) which can be used to write trace queries as XPath expressions for artifacts represented in XML format. The creation of queries by users is not an easy task. D3TraceView provides pre-defined questions, which can be extended in the tool.

5. CONCLUSION AND FUTURE WORK

In this paper, we presented a traceability visualization tool called D3TraceView that supports various visualization formats such as matrix, list, tree, radial, gauge, sunburst, table, and bar view. The tool is built using D3.js library and applies information visualization principles to present traceability relations. Currently, we are extending the tool with an editor to support the specification of queries. We are also evaluating the tool in terms of its usability and support for the various software development activities in the tool.

REFERENCES

[1] Bostock, M., Ogievetsky, V. and Heer, J. 2011. D3: Data-Driven Documents. In *Proceedings of the IEEE Trans. Visualization & Comp. Graphics.*

[2] Chen, X., Hosking, J. and Grundy, J. 2012. Visualizing traceability links between source code and documentation. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*.

[3] Cleland-Huang, J., Gotel, O., Hayes, J., Mäder, P. and Zisman, A. Software traceability: trends and future directions. *Proc. Future of Software Engineering*, 2014.

[4] Cysneiros, G. and Zisman, A. 2008. Traceability and completeness checking for agent-oriented systems. In *Proceedings of the 23rd Annual ACM Symposium on Applied Computing*.

[5] Cysneiros, G. and Lencastre, M.. Towards a Traceability Visualisation Tool. 2012. In *Proceedings of the 2012 Eighth International Conference on the Quality of Information and Communications Technology* (QUATIC '12).

[6] Gotel, O. and Finkelstein, A. 1994. An Analysis of the Requirements Traceability Problem. 1994. In *Proceedings of the First IEEE International Conference on Requirements Engineering* (ICRE'94).

[7] Gotel, O., Cleland-Huang, J., Zisman, A., Hayes, J., Dekhtyar, A., Mäder, P., Egyed, A., Grünbacher, P., Antoniol, G. and Maletic, J. 2012. Glossary of Traceability Terms (1.0). In Cleland-Huang, J., Gotel, O. and Zisman, A. editors, *Software and Systems Traceability*, Springer.

[8] Gotel, O., Cleland-Huang, J., Hayes, J., Zisman, A., Egyed, A., Grünbacher, P., Antoniol. G. 2012. The quest for Ubiquity: A roadmap for software and systems traceability research. In *Proceedings of the 2012 20th IEEE International Requirements Engineering Conference (RE)*.

[9] Heer, J., Bostock, M., and Ogievetsky, V. A tour through the visualization zoo. *Commun. ACM* 53, 6 (June 2010), 59-67.

[10] Heim, P., Lohmann, S., Lauenroth, K. and Ziegler, J. 2008. Graph based visualization of requirements relationships. In *Proceedings of the 2008 Requirements Engineering Visualization*.

[11] Howden, N., Rönnquist, R., Hodgson, A. and Lucas, A. 2001. JACK intelligent agents - Summary of an agent infrastructure. In *Proceedings of the 5th International Conference on Autonomous Agents*.

[12] Li, Y. and Maalej, W. 2012. Which traceability visualization is suitable in this context? a comparative study. In *Proceedings of the 18th International Conference on Requirements Engineering Foundation for Software Quality*.

[13] Mäder, P. 2013. Interactive Traceability Querying and Visualiation for Coping with Development Complexity. In *CoRR*.

[14] Mäder, P. and Cleland-Huang, J. 2013. A visual language for modeling and executing traceability queries. *Softw. Syst. Model.* 12, 3 (July 2013), 537-553.

[15] Maletic, J. and M. Collard, L. 2009. Tql: A query language to support traceability. In *Proceedings of the 5th Workshop on Traceability in Emerging Forms of Software Engineering*.

[16] Marcus, A., Xie, X. and Poshyvanyk, D. 2005. When and how to visualize traceability links? In *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*.

[17] Merten, T., Jueppner, D. and Delater, A. 2011. Improved Representation of Traceability Links in Requirements Engineering Knowledge using Sunburst and Netmap Visualizations. In *Proceedings of the 4th International Workshop on Managing Requirements Knowledge*.

[18] OpenUP. http://epf.eclipse.org/wikis/openup/

[19] Padgham, L. and Winikoff, W. 2004. *Developing Intelligent Agent Systems–A Practical Guide*, John Wiley & Sons.

[20] Thommazo, A., Malimpensa, G., Oliveira, T., Olivatto, G. and Fabbri, S. 2012. Requirements Traceability Matrix: Automatic Generation and Visualization. In *Proceedings of the 26th Brazilian Symposium on Software Engineering*.

[21] Voytek, J and Núnez, J. 2011. Visualizing Non-Functional Traces in Student Projects in Information System and Service Design. In *Proceedings of the Intl. Conf. Human Factors in Comp. Systems*.

[22] Winkler, S. 2008 On Usability in Requirements Trace Visualizations, In *Proceedings of the 2008 Requirements Engineering Visualization*.

[23] Yu, E. 1995. *Modelling Strategic Relationships for Process Reengineering*. PhD thesis, University of Toronto, Department of Computer Science, 1995.

Analyzing Complex Data in Motion at Scale with Temporal Graphs

Thomas Hartmann*, Francois Fouquet*, Matthieu Jimenez*, Romain Rouvoy† and Yves Le Traon*
*University of Luxembourg, Luxembourg, firstname.lastname@uni.lu
†Univ. Lille / Inria / IUF, France, romain.rouvoy@inria.fr

***Abstract*—Modern analytics solutions succeed to understand and predict phenomenons in a large diversity of software systems, from social networks to Internet-of-Things platforms. This success challenges analytics algorithms to deal with more and more complex data, which can be structured as graphs and evolve over time. However, the underlying data storage systems that support large-scale data analytics, such as time-series or graph databases, fail to accommodate both dimensions, which limits the integration of more advanced analysis taking into account the history of complex graphs, for example. This paper therefore introduces a formal and practical definition of temporal graphs. Temporal graphs provide a compact representation of time-evolving graphs that can be used to analyze complex data in motion. In particular, we demonstrate with our open-source implementation, named GREYCAT, that the performance of temporal graphs allows analytics solutions to deal with rapidly evolving large-scale graphs.**

***Index Terms*—Data analytics, graph databases, large-scale graphs, time-evolving graphs**

I. INTRODUCTION

The data deluge induced by large-scale distributed systems has called for scalable analytics platforms. Modern analytics solutions succeed to understand and predict phenomenons in a large diversity of software systems, from social networks to Internet-of-Things platforms. Graphs are increasingly being used to structure and analyze such complex data [1], [2], [3]. However, most of graph representations only reflect a snapshot at a given time, while reflected data keeps changing as the systems evolve. Understanding temporal characteristics of time-evolving graphs therefore attracts increasing attention from research communities [4]—*e.g.*, in the domains of social networks, smart mobility, or smart grids [5].

Yet, state-of-the-art approaches fail to provide a scalable solution to effectively support time in graphs. In particular, existing approaches represent time-evolving graphs as sequences of full-graph snapshots [6], or they use a combination of snapshots and deltas [7], which requires to reconstruct a graph for a given time, as depicted in Figure 1. However, full-graph snapshots tend to be expensive in terms of memory requirements, both on disk and in-memory. This overhead becomes even worse when data from several snapshots need to be correlated, which is the case for most of advanced analytics [5], [6], [8]. Another challenging issue related to snapshots relates to the snapshotting frequency: regardless of changes, for any change in the graph, or only for the major changes, which results in a tradeoff between duplicating data and feeding analytics with up-to-date metrics. This is crucial when data evolves rapidly and at different paces for different elements in the graph, like it is for example the case with sensor data in domains like the *Internet-of-Things* (IoT) or *Cyber-Physical Systems* (CPS) [5].

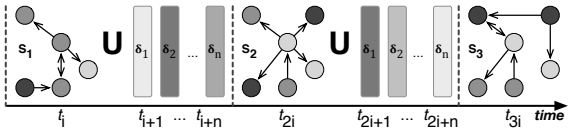


Fig. 1. Snapshots (S_i) and deltas (δ_n) of a time-evolving graph

An alternative to snapshotting consists in combining graphs with time series databases [9], by mapping individual nodes to time series. However, this becomes quickly limited when large parts of the graph evolve over time, inducing multiple time queries to explore the graph. Moreover, the description and the evolution of relationships among the nodes of the graph are rather hard to model within a time series database.

In this paper, we therefore introduce a novel *temporal graph* data model and storage, which allow analytics platforms to represent time-evolving graphs in an efficient manner. Most notably, our approach completely adopts a radically new approach by using an innovative, node-scale, and on-demand cloning approach. In temporal graphs, each node can evolve independently in time, while graphs are traversed for arbitrary timestamps.

We demonstrate with our open-source implementation of temporal graphs, named GREYCAT[1], that the performance of this data model allows analytics solutions to deal with rapidly evolving large-scale graphs. We compare our solution with a complete snapshotting approach and a combination of graph and time series databases.

DOI: 10.18293/SEKE2017-048

[1]https://github.com/datathings/greycat

The remainder of this paper is organized as follows. In Section II, we first formalize the semantics of our temporal graph data model. Then, in Section III, we present and discuss implementation details of this data model within GREYCAT, our open-source graph framework. We thoroughly evaluate the temporal aspects of GREYCAT in Section IV, before discussing the related work in Section V and concluding the paper in Section VI.

II. DEFINITION OF THE TEMPORAL GRAPH SEMANTICS

A graph G is commonly defined as an ordered pair consisting of a set V of nodes or vertices and a set E of edges: $G = \{V, E\}$. We define a slightly different semantics for our temporal graph by distinguishing between a node and its *state*. We define a node as a *conceptual identifier* that is mapped to its state, which we refer to as *state chunk*. It contains the values of all attributes and edges that belong to a node. Attributes are typed according to one of the following primitive types: `int`, `long`, `double`, `string`, `bool`, and `enumeration`. Formally, we define a state chunk as: The state chunk c of a node n is $c_n = (A_n, R_n)$, where A_n is the set of attribute values of n and R_n is the set of relationship values from n to other nodes. Unlike other graph models (*e.g.*, Neo4J [10]), ours does not support edge attributes (like the OO model). However, any edge attribute can be modeled leveraging an intermediate node. Next, we define the function $read(n)$ to resolve the state chunk of a node. We use this function to define a graph G as: $G = \{read(n), \forall n \in N\}$, where N is the set of nodes. Unlike common graph definitions, temporal graphs are not defined statically, but dynamically—*i.e.*, their are created as the result of the evaluation of the $read(n)$ function over all nodes.

The separation between the concept of a node and its state chunk is essential for our approach. First, it enables the implementation of a lazy loading mechanism—*i.e.*, by loading state chunks on-demand into main memory, while the graph is traversed. As further discussed in Section III, we build on key/value stores as storage backends for the temporal graphs. This mapping of a graph to keys and values is similar to what is proposed in [11]. Secondly, it allows to define different states for each node depending on the time. Therefore, we extend the previous definition of a graph with temporal semantics. We override the function $read(n)$ with $read(n, t)$, where $t \in T$ and T is a totally ordered sequence of all possible timepoints: $\forall t_i, t_j \in T : t_i \leq t_j \vee t_j \leq t_i$. Next, we extend the definition of a state chunk with a temporal version: $c_{n,t} = (A_{n,t}, R_{n,t})$, where $A_{n,t}$ and $R_{n,t}$ are the sets of resolved attributes and relationships, for the node n at time t. Then, we define a temporal graph as follows: $TG(t) = \{read(n, t), \forall n \in N\}, \forall t \in T$. Every node of the TG can evolve independently and, as timepoints can be compared, they naturally form a chronological order. We define that every state chunk belonging to a node in a TG is associated to a timepoint and can therefore be queried along this chronological order in a sequence $TP \subseteq T$. We call this ordered sequence of state chunks the *timeline* of a node. The timeline tl of a node n is defined as $tl_n = \{c_{n,t}, \forall t \in TP \subseteq T\}$. We define three node operations:

$insert(c_{n,t}, n, t)$: $(c \times N \times T) \mapsto void$ as the function that inserts a state chunk c in the timeline t of a node n, such as: $tl_n := tl_n \cup \{c_{n,t}\}$.

$read(n, t)$: $(N \times T) \mapsto c$ is the function that retrieves, from the timeline tl_n, and up until time t, the most recent version of the state chunk of n which was inserted at timepoint t_i:

$$read(n,t) = \begin{cases} c_{n,t_i} & \text{if } (c_{n,t_i} \in tl_n) \\ & \wedge (t_i \in TP) \wedge (t_i < t) \\ & \wedge (\forall t_j \in TP \rightarrow t_j < t_i) \\ \emptyset & \text{otherwise} \end{cases}$$

$remove(n, t)$: $(N \times T) \mapsto void$ is the function that removes a node n and the associated state chunks from the time t.

Based on these definitions, although timestamps are discrete, they logically define intervals in which a state chunk can be considered as *valid* within its timeline. When executing $insert(c_{n_1,t_1}, n_1, t_1)$ and $insert(c_{n_1,t_2}, n_1, t_2)$, we insert 2 state chunks c_{n_1,t_1} and c_{n_1,t_2} for the same node n_1 at two different timepoints with $t_1 < t_2$. We define that c_{n_1,t_1} is valid in the open interval $[t_1, t_2[$ and c_{n_1,t_2} is valid in $[t_2, +\infty[$. Thus, an operation $read(n_1, t)$ resolves $\emptyset$ if $t < t_1$, c_{n_1,t_1} when $t_1 \leq t < t_2$, and c_{n_1,t_2} if $t \geq t_2$ for the same node n_1. After executing $remove(n_1, t_3)$, $read(n_1, t)$ resolves $\emptyset$ if $t \geq t_3$. Since state chunks with this semantics have temporal validities, relationships between nodes also have temporal validities. This leads to *temporal relationships* between TG nodes and forms a natural extension of relationships in the time dimension. This temporal validity definition follows and extends our previous work [5]. It enables to transparently navigate inside the graph without considering time for every navigation step, by always loading the last valid version relative to the node the navigation started from.

III. GREYCAT: A TEMPORAL GRAPH IMPLEMENTATION

A. Mapping nodes to state chunks

The proposed temporal graph data model is a conceptual view of data to represent and analyze time-evolving complex data. Internally, we structure the data of a temporal graph as an unbounded set of *state chunks*. Therefore, we map the conceptual nodes (and relationships) of a temporal graph to *state chunks*. State chunks are the internal data

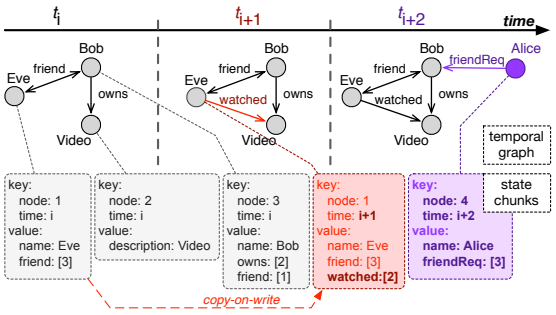


Fig. 2. Mapping of temporal graphs to state chunks

structures reflecting a temporal graph and, at the same time, also used for storing the temporal graph data to persistent storage. A state chunk contains, for every attribute of a node, the name and value of the attribute and, for every outgoing relationship, the name of the relationship and a list of identifiers of the referenced state chunks. Figure 2 depicts a concrete example of the mapping of nodes to state chunks, according to the semantic definitions of Section II.

At time t_i (start of the temporal graph), GREYCAT maps the nodes and the relationships to 3 state chunks: Bob, Eve, and Video. At time t_{i+1}, the graph evolves to declare a relationship watched from Eve to Video. Since this evolution only affects Eve, GREYCAT only creates an additional state chunk for Eve at time t_{i+1} by cloning and modifying the previous version of Eve's state chunk (using copy-on-write). All other nodes are therefore kept unchanged at time t_{i+1}. At time t_{i+2}, Bob meets Alice, who sends a friend request to Bob. As only Alice's status changes, GREYCAT only needs to create a new state chunk for Alice from time t_{i+2}.

In this example, the graph contains 10 different conceptual nodes and 12 relationships (counting each bidirectional relation as two edges) and evolves along 3 different timestamps, but GREYCAT only stores 5 state chunks to model the whole temporal graph. Whenever the temporal graph is explored, the correct state chunks are retrieved depending on the requested time.

B. Lazy-loading state chunks

State chunks are the units of storage in GREYCAT. They are stored on disk and loaded into main memory while the graph is explored or when nodes are explicitly requested. The loading of state chunks is achieved lazily by GREYCAT, because only attributes and sets of identifiers are loaded. This theoretically allows GREYCAT to process temporal graphs of unbounded size even with restricted main memory. For persistent storage of state chunks, we rely on key/value stores by using the tuple of $(node, time)$ as key and the state chunk as value. We serialize chunk states into Base64 encoded `blobs`. This format reduces the required interface to insert, read, and remove state chunks to a persistent data store. It allows to use different storage backends depending on the requirements of an application: from in-memory key/value stores up to distributed and replicated NoSQL databases. For fault tolerance and concurrency, we use a per-node lock policy and rely on the underlying storage technology to ensure concurrency and distribution. With a node we also lock the index structures associated to it (cf. Section III-D). Another consistency approach, based on consistent global checkpoints, is discussed in [12]. All clients reading the node (at the same time) see updates on it. This mapping approach copies state chunks only on-demand—*i.e.*, copy-on-write—and ensures efficient read and write operations at any point in time. Basically, it enables analytics algorithms to be executed with constant memory requirements.

C. Caching state chunks

Despite of having a positive effect on memory, lazy loading significantly increases input/output (I/O) operations. Therefore, we rely on caching mechanisms to reduce I/O operations, which means that some state chunks are kept in memory, based on their probability of being reused. In contrary to pure key/value storages, our state chunks have semantic relationships, which can be leveraged by the caching mechanism. For instance, if contiguous timepoints of the same node are loaded, the temporal index has a high probability to be reused. We build our caching mechanism as a *Least-Recently-Used* (LRU) cache, where each read operation is taken into account for the cache victim eviction computation. To reflect semantic relationships, we count in temporal index LRU scores: the number of state chunks using them. This way, we are giving chances for temporal indexes to be reused to access other timepoints.

D. Indexing state chunks using red-black trees

Temporal graphs can be composed by highly volatile nodes, characterized by a very long timeline of millions of timepoints. Such volatile nodes are for example needed to store sensor data collected by IoT devices. In order to efficiently execute temporal queries, these timepoints must be indexed. For scalability reasons, GREYCAT relies on a lazy loading mechanism for retrieving nodes [5] . In a similar way, we define temporal indexes with a semantic to avoid loading millions of timepoints, *i.e.*, the full temporal index, if only parts of the timepoints are actually matching a query. Queries in GREYCAT are always precise.

Temporal characteristics of timepoints—*i.e.*, regularity, periodicity—make some indexing approaches more suitable than others [13]. For non-monotonic measurements, balanced trees offer one of the best compromises between read and insert performance. In particular, *Red-Black Trees* (RBT) are one of the most adopted structures to index non-monotonic time-series [14]. However, RBTs are in-memory structures that cannot be partially loaded due to

their balanced hierarchy. To workaround this limitation, we defined an adaptive multi-layer on top of RBTs in order to split temporal indexes in pieces that can be lazily loaded.

Our approach, named *Adaptive Multi-Layered Red-Black Tree* (AMT), is based on the same balancing principle than RBTs are, but at a coarser granularity to support efficient lazy loading. Therefore, we defined the notion of a *supertree* that can index fixed-sized subtrees, based on their oldest timepoint. For every read and insert operation, the supertree is first loaded and used to locate the relevant subtrees, which in turn are used to load an updated index. In addition, to compact the size of supertrees, we define an adaptive strategy to reconfigure the size of subtrees according to the size of the supertree. This adaptive scattering mechanism offers an efficient approach for short timelines to use lazy loading, which is automatically relaxed for longer ones, as illustrated by Algorithm 1.

Algorithm 1. Inserting t_i in the AMT for node n

```
procedure INSERT(n,t_i)
    tr_sup ← LOADTREE(n, t_0)
    t_r ← CLOSESTTIME(tr_sup, t_i)
    tr_sub ← LOADTREE(n, t_r)
    if SIZE(tr_sup) < step_1 then
        max ← step_1
    else if SIZE(tr_sup) < step_2 then
        max ← step_2
    else
        max ← step_3
    end if
    if SIZE(tr_sub) < max then
        INSERT(tr_sub, t_i)
    else
        ⟨tr_left, tr_right⟩ ← SPLIT(tr_sub)
        if t_i > LOWEST(tr_right) then
            INSERT(tr_right, t_i)
        else
            INSERT(tr_left, t_i)
        end if
    end if
end procedure
```

IV. Evaluation of GreyCat

In this section, we evaluate our reference implementation of the temporal graph model against two potential alternative implementations: *i)* a plain graph stored in a time series database and *ii)* a plain graph versioned with checkpoints. More specifically, we focus on read and write throughput, the elementary operations of data analytics.

A. Experimental Protocol

All the reported experiments were executed on a Linux server with a 12-core Intel Xeon E5-2430 processor with 128 GB of memory. Experiments have been executed 10 times and the reported numbers refer to mean values. Experiments are made available on GitHub[2] for the sake of reproducibility. To compare these approaches, we consider a synthetic graph of $100,000$ nodes. The generated graph corresponds to a k-ary tree where each node includes a unique identifier, an integer value, a character and, if necessary, a link to the parent node as well as links to the children nodes. In a second step, we update $15\,\%$ of the nodes of the graph repetitively, until reaching an history of $5,000$ changes. This ratio is extracted from a smart grid topology generator, which is based on a realistic smart grid dataset [15]. A node change consists in randomly setting a new value and a new character to the node. Our evaluation aims at testing the scalability of each solution for a growing temporal graph history. Therefore, we considered the following *Key Performance Indicators* (KPI): *read throughput* and *write throughput*. Throughput indicators are reported in *nodes read or written per second*.

[2]https://github.com/electricalwind/greycatBench

B. Alternative Implementations of Temporal Graphs

We compare GreyCat with 2 alternative implementations of temporal graphs: using time series and using checkpoints.

1) Storing Graphs as Time Series: To build this candidate solution we leveraged a plain graph whose values are versioned in InfluxDB [16] (version 1.1.2). InfluxDB is one of the newest and fastest time series databases that received much attention lately. This category of databases is heavily used for data mining and forecasting [17], [18]. While many time series databases provide interesting features, like SQL-like query languages, their data model is essentially flat—usually integer or double values—and does not support complex relationships between data. In a time series based temporal graph, each node of the graph maintains its own corresponding time series. Thus, to retrieve the version of a node at time t, one must first fetch its time series and then retrieve the value at that time. However, a major limitation of such an approach in the context of temporal graphs is that it is not possible to directly store relations. In our evaluation, we chose to represent the relationship of a node A to a node B and C as a field in node A, containing the identifiers of B and C.

2) Storing Graph with Checkpoints: Another possibility—and the most common one [7], [8]—is to take a snapshot of the graph at regular intervals. A snapshot is a complete copy of the full graph. Thus, conversely to time series, to retrieve a node at time t, one will first have to load the closest previous snapshot and then find the node in the graph of this snapshot. Despite being very simple, such solution comes at the price of *i)* redundancy, *ii)* possible missing values if a node were updated twice along one interval, and *iii)* the impossibility to create new nodes in the past. Techniques exist to mitigate the expensive cost of a full copy, such as chained immutable trees, where every update consists

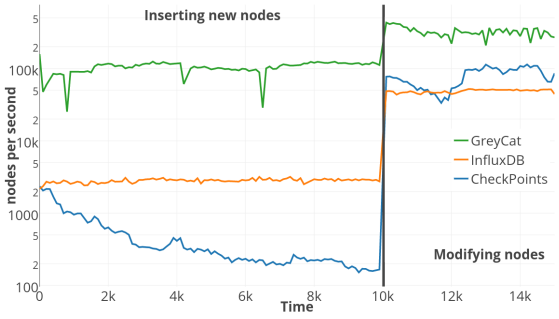


Fig. 3. Write throughput when creating and updating nodes

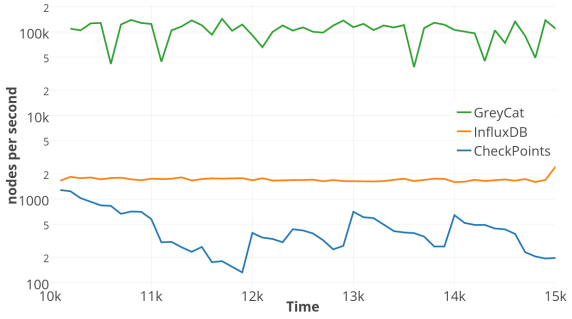


Fig. 4. Read throughput: *sum of children*

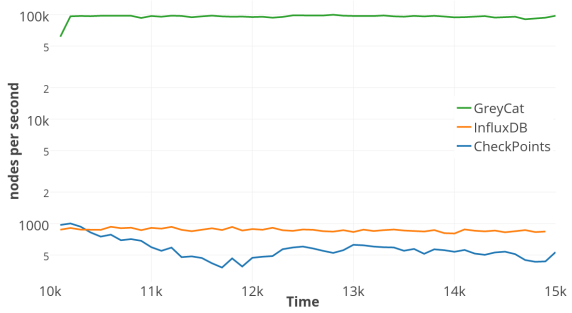


Fig. 5. Read throughput: *string building*

in a wrapper of the previous version plus a delta. Such techniques are used to implement transactional storages. In our evaluation, we use the *checkpoints* mechanism offered by RocksDB [19], to store graph data with a minimum amount of redundancy, thanks to the use of hard links from a new version to the previous stored one.

C. Empirical Evaluation

To measure the efficiency of each solution, we consider 3 steps. First, we load the temporal graph and explore its history—the throughput is measured for every step of the graph evolution. Then, we compute the recursive sum of all children's integer values of a node recursively at a given time. This requires to read a large number of nodes (up to a tenth of the total number of nodes) and a large number of graph traversals. Finally, we simulate the construction of a synthetic state vector by building a string with the character value of the n^{th} child of a node recursively at a given time. This requires fewer reads (up to log_{10} of the total number of nodes), but emphasizes on the ability to handle large relationships. These benchmarks are evaluated against the 3 temporal graph approaches, which adopt different design choices. To improve the overall readability, graphs have been smoothen by gathering measure per group of 100 timestamps and then averaged.

1) Write Throughput: Fig. 3 demonstrates that GreyCat outperforms the other solutions, with a write throughput close to $100,000$ nodes per second for step 1 and $400,000$ for step 2. The differences for the two steps can be explained by the fact that creating a node is costlier than updating one (for all solutions). Another interesting observation is that snapshotting performs worse than InfluxDB for large-scale graphs for the insertion, but slightly better for the update step.

2) Read Throughput: Fig. 4 and 5 depict the read throughput when trying to access nodes in one of their previous states for the two use cases presented above. Note that the result starts at $10,000$ as we wait for step 2 to start to perform read measurements.

Similarly to the write throughput, GreyCat performs significantly better than the two other solutions in both scenarios for reading, with around $100,000$ nodes per second. Time series are also stable over time in both cases, but as expected, the read throughput of traversing specific relationships, is twice slower than traversing all the nodes of the relationship. In the case of snapshots, performance is quite similar in both situations, but decreases over time.

V. Related Work

The need to deal with temporal data has been discussed across several research communities. Early works in database communities [20], [21] delivered formal semantics for historical relational databases. Some of these temporal features are integrated into SQL:2011 or ISO/IEC 9075:2011 [22]. The necessity to reason about time-evolving data has also been discussed in the area of the semantic web, *e.g.*, Motik [23]. With the emergence of big data and the IoT, temporal aspects of data, in form of time series databases [16], [24], gained again visibility in research communities. However, the data model of time series databases is essentially flat and does not support complex relationships between data.

Temporal graph processing frameworks go a step further and consider time-evolving graphs. Chronos [7], and its extension ImmortalGraph [8], are storage and execution engines for graph computations on temporal graphs. They define a temporal graph as a sequence of graph snapshots at specific points in time. To store temporal graph data on disk, they use so-called snapshot groups. A

snapshot group is valid for a time interval and comprises a complete snapshot for the beginning of the interval and a number of deltas until the end of the interval. GRAPHTAU [6], *Historical Graph Store* (HGS) [25], G* [26], and KINEOGRAPH [27] are other temporal graph processing frameworks. They all represent time-evolving graphs as series of consistent graph snapshots. Furthermore, for none of these solutions the source code is available. Some of them optimize storage by using a combination of complete snapshots at specific timepoints and deltas in-between these [7]. Nonetheless, in some form or another, data models of existing approaches represent time-evolving graphs as sequences of full graph snapshots. This comes with severe limitations: First of all, full-graph snapshots are expensive in terms of memory requirements (both on disk and in-memory). Secondly, for every small change in the graph it would be necessary to snapshot the graph (and/or the delta) to keep track of the change history. Thirdly, the continuous semantics of time is lost by the discretisation in snapshots. Thus, navigating in the time and space dimensions of the graph is problematic, which complicates analytics algorithms.

VI. CONCLUSION AND DISCUSSION

In this paper, we presented a temporal graph data model and GREYCAT, its open source reference implementation. Most notably, our approach is able to model large-scale, time-evolving graphs without relying on snapshotting, like the current state-of-the-art does [7], [8]. Moreover, GREYCAT is one of the only open source frameworks for time-evolving graphs. We demonstrated that our temporal graphs pave the way for analyzing complex data in motion at scale. In particular, we illustrate that this data model is especially efficient when analyzing large-scale graphs with partial changes along time, which is typical for many real world analytics [5], [6].

REFERENCES

[1] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in *Proc. of the 2010 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD'10, 2010.

[2] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: A framework for machine learning and data mining in the cloud," *Proc. VLDB Endow.*, vol. 5, no. 8, Apr. 2012.

[3] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "Powergraph: Distributed graph-parallel computation on natural graphs," in *Proc. of the 10th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'12, 2012.

[4] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: Densification laws, shrinking diameters and possible explanations," in *Proc. of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, ser. KDD'05, 2005.

[5] T. Hartmann, F. Fouquet, G. Nain, B. Morin, J. Klein, and Y. L. Traon, "Model-based time-distorted contexts for efficient temporal reasoning," in *Proc of the 26th International Conference on Software Engineering and Knowledge Engineering*, 2014.

[6] A. P. Iyer, L. E. Li, T. Das, and I. Stoica, "Time-evolving graph processing at scale," in *Proc. of the 4th International Workshop on Graph Data Management Experiences and Systems*, ser. GRADES'16, 2016.

[7] W. Han, Y. Miao, K. Li, M. Wu, F. Yang, L. Zhou, V. Prabhakaran, W. Chen, and E. Chen, "Chronos: A graph engine for temporal graph analysis," in *Proc. of the 9th European Conference on Computer Systems*, ser. EuroSys'14, 2014.

[8] Y. Miao, W. Han, K. Li, M. Wu, F. Yang, L. Zhou, V. Prabhakaran, E. Chen, and W. Chen, "Immortalgraph: A system for storage and analysis of temporal graphs," *Trans. Storage*, Jul. 2015.

[9] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proc. of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, ser. DMKD'03, 2003.

[10] J. J. Miller, "Graph database applications and concepts with neo4j," in *Proc. of the Southern Association for Information Systems Conference*, vol. 2324, 2013.

[11] B. Shao, H. Wang, and Y. Li, "Trinity: A distributed graph engine on a memory cloud," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '13. New York, NY, USA: ACM, 2013, pp. 505–516.

[12] Y.-M. Wang, "Consistent global checkpoints that contain a given set of local checkpoints," *IEEE Transactions on Computers*, vol. 46, no. 4, pp. 456–468, Apr 1997.

[13] R. Elmasri, Y.-J. Kim, and G. T. Wuu, "Efficient implementation techniques for the time index," in *Proc. 7th International Conference on Data Engineering*. IEEE, 1991.

[14] L. J. Guibas and R. Sedgewick, "A dichromatic framework for balanced trees," in *Foundations of Computer Science, 1978., 19th Annual Symposium on*. IEEE, 1978.

[15] T. Hartmann, F. Fouquet, J. Klein, Y. L. Traon, A. Pelov, L. Toutain, and T. Ropitault, "Generating realistic Smart Grid communication topologies based on real-data," in *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*.

[16] "influxdb: Time-Series Data Storage," https://influxdata.com/time-series-platform/influxdb.

[17] E. Keogh, S. Lonardi, and B. Y.-c. Chiu, "Finding surprising patterns in a time series database in linear time and space," in *Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '02. New York, NY, USA: ACM, 2002, pp. 550–556.

[18] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," in *Proc. of the 1994 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '94. New York, NY, USA: ACM, 1994.

[19] "Use Checkpoints for Efficient Snapshots," http://rocksdb.org/blog/2015/11/10/use-checkpoints-for-efficient-snapshots.html.

[20] J. Clifford and D. S. Warren, "Formal semantics for time in databases," *ACM Trans. Database Syst.*, vol. 8, no. 2, Jun. 1983.

[21] E. Rose and A. Segev, "Tooa: A temporal object-oriented algebra," in *Proc. of the 7th European Conference on Object-Oriented Programming*, ser. ECOOP'93, 1993.

[22] K. Kulkarni and J.-E. Michels, "Temporal features in sql:2011," *SIGMOD Rec.*, vol. 41, no. 3, Oct. 2012.

[23] B. Motik, "Representing and querying validity time in rdf and owl: A logic-based approach," *Web Semant.*, vol. 12-13, Apr. 2012.

[24] "OpenTSDB: The Scalable Time Series DB," http://opentsdb.net.

[25] U. Khurana and A. Deshpande, "Storing and analyzing historical graph data at scale," *CoRR*, vol. abs/1509.08960, 2015.

[26] A. G. Labouseur, J. Birnbaum, P. W. Olsen, Jr., S. R. Spillane, J. Vijayan, J.-H. Hwang, and W.-S. Han, "The g* graph database: Efficiently managing large distributed dynamic graphs," *Distrib. Parallel Databases*, vol. 33, no. 4, Dec. 2015.

[27] R. Cheng, J. Hong, A. Kyrola, Y. Miao, X. Weng, M. Wu, F. Yang, L. Zhou, F. Zhao, and E. Chen, "Kineograph: Taking the pulse of a fast-changing and connected world," in *Proc. of the 7th ACM European Conference on Computer Systems*, ser. EuroSys'12, 2012.

MAGICIAN: Model-based design for optimizing the configuration of data-centers

Pablo C. Cañizares ★ Alberto Núñez ★ Juan de Lara ☆

★ Universidad Complutense de Madrid
Spain

☆ Universidad Autónoma de Madrid
Spain

Abstract

Designing data-centers that provide an acceptable cost-performance ratio is challenging. Generally, a wide spectrum of components must be previously analyzed, such as the kind of applications to be executed in the data-center, computing/storage requirements and the network topology, among others. Since each one of these components has a direct impact on the overall system performance, the design process is complex and difficult, which usually requires the intervention of an expert.

We propose a model-based approach to design data-centers. For this purpose, we have created a meta-model that describes the structure of data-center models. Then, a set of expert rules can be used to detect sub-optimal configurations, and (in some cases) correct the design. Data-center models can be simulated, to assess their performance and scalability, for which we use a code generator into the SIMCAN tool. We have implemented our approach as an Eclipse plugin, and illustrate the usefulness of some expert rules by showing the efficiency and scalability gains of the optimized model with respect to the original one.

1 Introduction

During the last decade, most efforts in scientific applications were focused in obtaining the best possible performance, exploiting the system resource usage both in supercomputers and commodity clusters.

Due to the high number of inter-related parameters that have a direct impact on the overall performance, building a system that provides the maximum performance for a given application is a very complex task. Designing and configuring a data-center that properly exploits the system resource usage may be a feasible task for an expert [1]. However, when the data-center is designed by a non-expert, it may provide an overall performance far from the expected one. Generally, a misconfiguration of the system architecture or a wrong choice of hardware resources, may lead to obtaining a poor performance.

Usually, the first step before deploying the data-center in a production environment consists in modelling and simulating its underlying architecture. Thus, the obtained results from the simulation are used to polish and improve the initial design. Unfortunately, the number of possible configurations is extremely large, making unpractical to model and simulate all of them.

In this paper we propose `MAGICIAN`, an approach that aids designers to optimize the configuration of data-centers. The main objective of `MAGICIAN` is to identify possible inconsistencies in the initial design of the data-center and to suggest feasible corrections. Thus, a reduced number of data-centers designs are generated, which can be simulated to analyze which one provides the best results.

The approach is based on model-driven engineering (MDE) [2]. We propose a meta-model for data-centers, so that data-center configurations are expressed as instances of such meta-model. We provide a library of expert knowledge rules to detect misconfigurations and suggest improvements on the design. Finally, we support the simulation of the data-center configuration to assess properties like scalability, and detect possible bottlenecks. The simulation is performed by generating code for the SIMCAN tool [3].

The rest of this paper is organized as follows. Section 2 provides an overview of the proposed approach. Section 3 describes in detail the principal components of `MAGICIAN`. Next, Section 4 presents performance experiments that show the usefulness of our approach. Section 5 presents related work, and Section 6 ends with the conclusions and future work.

2 Overview

In this section, we describe the `MAGICIAN` approach for optimizing the configuration of data-centers. The overall scheme is shown in Figure 1.

Our architecture represents data-center designs as models conformant to a meta-model, and includes two optimization loops. The first one is based on expert rules, which encode knowledge on typical good configurations. The second is based on simulation, with which one can analyse aspects like efficiency or scalability of the data-center models.

The structure of data-centers, and the integrity constraints that valid data-center models should obey, have been captured through a meta-model. The designer will

DOI reference number: 10.18293/SEKE2017-108

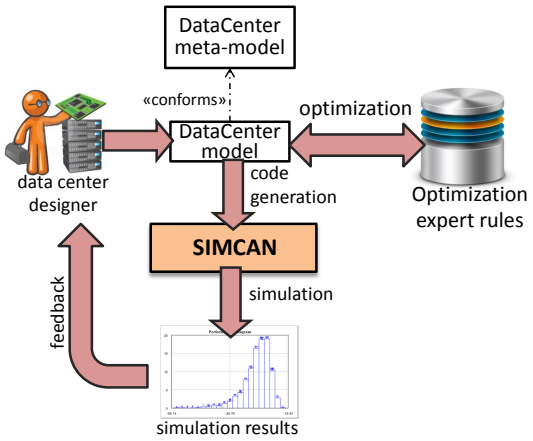


Figure 1. Working scheme of our approach

be able to create models that *conform* to such meta-model. That is, these models use the types and relations defined in the meta-model, and satisfy the integrity constraints. This meta-model will be explained in Section 3.1.

We have created a library of expert rules, containing optimization patterns and idioms, typically followed by good data-center designs. These rules detect parts of the model that are amenable to optimization, signalling potential deficiencies in the model. Moreover, some of these rules contain quick fixes, which modify the design to improve some suboptimal aspect of the model. Technically, these rules have been implemented using the Epsilon Validation Language (EVL) [4]. They will be detailed in Section 3.2.

We also enabled the evaluation of the data-center model through simulation. This way, we have built a code generator that produces code to be executed by the SIMCAN simulation tool [3]. The results of the simulation can be used by the designer to find problems in the design, such as bottlenecks, to further improve it. The approach to code generation, and details on how the simulation is performed are given in Section 3.3.

3 Model-based simulation of data-centers

In this section we explain the three main building blocks of our approach: the data-center meta-model (see Section 3.1), the expert rules and quick fixes (see Section 3.2), and the code generation and simulation (see Section 3.3).

3.1 The data-center meta-model

Figure 2 shows a simplified version of the data-center meta-model. The *DataCenter* meta-class is the root class, which contains the main elements of a real data-center, such as those relating with both computational and networking aspects.

The computing elements are divided in two types. The first type corresponds to the *Node* meta-class, which represents a single computational node. The first 3 attributes define the CPU processor, where *CPU_Sockets*, *CPU_Cores* and *CPU_Speed* represent the number of CPUs, the number of cores of each CPU and the CPU speed (measured in MIPS), respectively. The next 3 attributes are related with memory features, where *RAM_slots*, *RAM_Size* and *RAM_Frequency* represent the number of memory modules, the total size of each module (measured in GBytes) and the frequency of the memory (measured in Mhz), respectively. The next 4 attributes refer to storage aspects, where *Disk_Slots*, *Disk_Size Disk_RBandwidth* and *Disk_WBandwidth* are the number of disks, the size of each disk (measured in GBytes) and the read and write bandwidth of the storage system (measured in Gbps), respectively. The last attribute, *isComputingNode*, denotes if a given node is a computing node or a storage node. The second type of computing elements corresponds to the *Rack* meta-class. A rack represents a structure that contains multiple computing elements. In this case, the rack consists of a set of *Boards*, where each board contains a number of nodes that is determined by the attribute *Nodes_per_board*.

The network is defined by 2 elements. The *Network* meta-class represents the communication network of the data-center, where *Bandwidth*, *Latency*, and *ErrorRatio* define the data transfer rate (measured in Gbps), the latency (measured in μs) and the error ratio of the network, respectively. The *Switch* meta-class represents a resource used to communicate the different computing elements of the data-center through the communication network, where *MTU* and *NumPorts* are the maximum transmission unit and the number of ports of the switch, respectively.

Finally, the *Repository* meta-class represents the data-center repository, which provides a wide collection of networking and computational components to model, with a high level of detail, a complete data-center.

Figure 3 shows an example of a data-center model which conforms the proposed meta-model. This model is inspired by a real IBM data-center configuration, which consists of 1 *IBM Flex* rack and 1 *IBM v7000 Storage* rack. These racks are interconnected using a 40/10 Gigabit Ethernet communication network and a *SAN42B-R extension switch*.

The *IBM Flex* rack consists of 6 *Flex System Enterprise Chassis* boards, where each board contains 14 *IBM Flex System p460* computing nodes. These boards consist of 4 CPUs with 8 cores, reaching a speed of 317.900 MIPS. The memory system consists of 32 slots, which contain 64 GB of RAM. Finally, the storage consists of two disks of 2 TBytes.

The *IBM Storage* rack consists of 6 *Flex System Chassis Storage* boards, where each board contains 14 *IBM v7000* storage nodes. Each node consists of 2 CPUs with 4 cores,

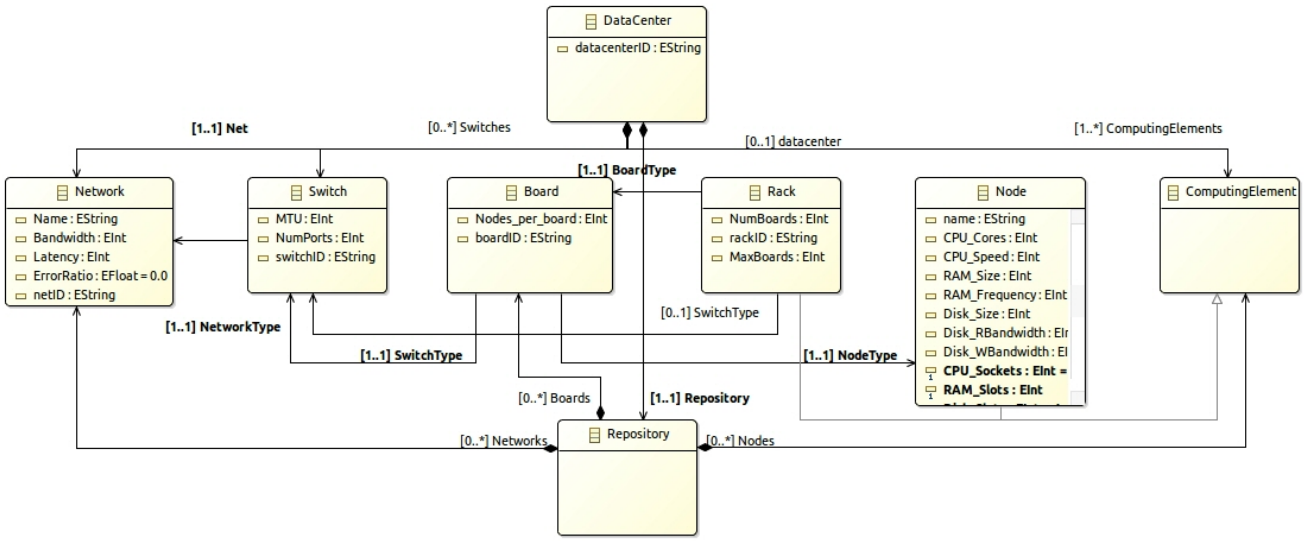


Figure 2. The data-center meta-model (excerpt)

reaching an speed of 200.000 MIPS, 16 GB of RAM and 16 hard disks with a total storage of 10 TBytes.

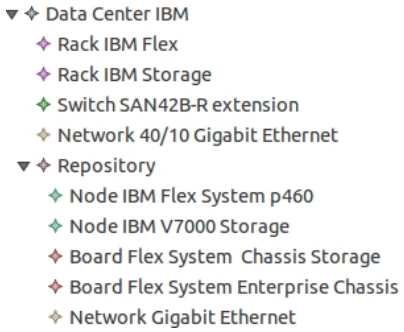


Figure 3. Example of a data-center model

3.2 Encoding expert rules and heuristic quick fixes

In order to support the user during the data-center design process, we have included a library of optimization rules based on data-center experts knowledge. These expert rules aid the user to solve possible design issues, which in most cases, hamper the overall data-center performance. However, expert rules must be designed and provided by an expert user, who must decide whether these are suitable to cover the requirements of the systems under study. The library consists of several rules focused on analysing different features of the data-center components, such as CPU processors, the memory system, storage and connectivity, among others. The main goal of these rules is to find inconsistencies in data-center models and to provide relevant information to fix them. For the sake of simplicity, in this paper we have described a sample of three rules from the complete library.

Listing 1 shows the expert rule *CoresVsStorageNodesRatio* encoded in EVL. This rule analyses the ratio between the number of storage nodes and the number of CPUs of a data-center. The main objective of this rule is to avoid system bottlenecks caused by a reduced number of storage nodes. In this case, if the available storage nodes are not able to provide the required performance, a message to modify the current design is shown. As can be seen, the rule is applied on the context of DataCenter objects (line 1 of the listing). It is made of a check section (lines 5–11), which evaluates a certain condition on the model, and a message part, which is presented to the designer if the check part returns true.

```
1  context DataCenter
2  {
3    critique CoresVsStorageNodesRatio
4    {
5      check{
6        var storageNodes: Integer;
7        var totalCores: Integer;
8        storageNodes = self.calculateStorageNodes();
9        totalCores = self.calculateTotalCores();
10       return storageNodes*40 >= totalCores;
11     }
12     message: 'The number of storage nodes must be increased, there
               exist a high number of cores in comparison with the number of
               storage node which can act as bottleneck'
13   }
14 }
```

Listing 1. Data-center topology optimization rule encoded in EVL

Listing 2 shows two expert rules based on the analysis of two network features, bandwidth and latency. In this case,

these rules check that these features range in a determined interval. If some of these features is out of the range, the system provides a quick-fix method to alleviate the issue. Quick fixes are specified in the fix section of the rules (lines 7–9 and 16–19). Figure 4 shows how such quick-fix is presented to the user.

```
1  context Network
2  {
3    critique NetBandwidth
4    {
5      check: self.Bandwidth>=10 and self.Bandwidth <=100
6      message: 'Network '+ self.Name + ': Bandwidth is usually ranged in
             [10−100]. Some of the most used configuration is 40'
7      fix {
8        title : "Set Bandwidth" + self.Name + " Bandwidth to 40"
9        do { self.Bandwidth = 40; }
10     }
11   }
12   critique NetLatency
13   {
14     check: self.Latency>=20 and self.Latency <=2000
15     message: 'Network '+ self.Name + ': Latency is usually ranged in
             [20−2000]. Some of the most used configuration is 200'
16     fix {
17       title : "Set Latency" + self.Name + " Latency to 40"
18       do { self.Latency = 200;}
19     }
20   }
21 }
```

Listing 2. Network optimization rules encoded in EVL

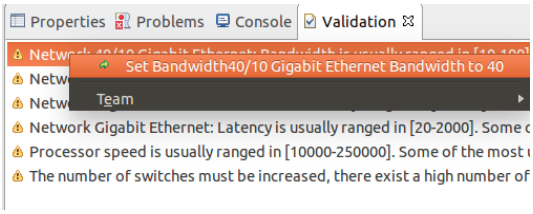


Figure 4. Example of network quick-fix

3.3 Code generation and Simulation

Once the data-center has been modelled, it can be simulated, to analyse its efficiency, in terms of scalability and performance. In this paper, we use the SIMCAN simulation platform to represent and simulate the behaviour of data-centers [3]. We have created a code generator that transforms the designed model into the required configuration files to perform the simulation.

Listing 3 shows an extract of the generated data-center topology, written in the NED language. The first line represents the name of the data-center, the next 3 lines refers to the different resources that compose the environment, such as switch, storage and computing nodes, respectively. Finally, the lines 6-8 show how the storage and computing elements are connected through the communication network, using the switch component.

```
1  network IBM{
2    switch_0:EtherSwitch;
3    rCmp1_IBM_Flex_Rack:Rack;
4    rSto0_StorageRack:Rack;
5
6    for i=0..5 {
7      rCmp1_IBM_Flex_Rack.ethg++ <−> Eth10M_channel <−>
           switch_0.ethg++;
8      rSto0_StorageRack.ethg++ <−> Eth10M_channel <−> switch_0.
           ethg++;
9    }
10 }
```

Listing 3. Example of data-center topology in SIMCAN written in NED language (excerpt)

Listing 4 shows an excerpt of a generated data-center configuration file. This portion of the configuration file configures the computing rack illustrated in Figure 3. It is important to remark that the symbol $*$ refers to a wildcard that represents all the elements in the referenced structure. For example, the lines 3 and 11 refer to the configuration of the network for all the boards in the rack.

```
1  IBM.rCmp1_IBM_Flex_Rack.numBoards = 6
2  IBM.rCmp1_IBM_Flex_Rack.nodesPerBoard = 14
3  IBM.rCmp1_IBM_Flex_Rack.nodeBoard[*].channelType = "Eth10M"
4  IBM.rCmp1_IBM_Flex_Rack.nodeBoard[*].node[*].cpuModule.CPUcore
       [*].speed = 79475
5  IBM.rCmp1_IBM_Flex_Rack.nodeBoard[*].node[*].bsModule[*].disk.
       readBandwidth = 650.0Mbps
6  IBM.rCmp1_IBM_Flex_Rack.nodeBoard[*].node[*].bsModule[*].disk.
       writeBandwidth = 420.0Mbps
7  IBM.rCmp1_IBM_Flex_Rack.nodeBoard[*].node[*].osModule.memory.
       size = 2.0GiB
8
9  IBM.rSto0_StorageRack.numBoards = 1
10 IBM.rSto0_StorageRack.nodesPerBoard = 1
11 IBM.rSto0_StorageRack.nodeBoard[*].channelType = "Eth10M"
12 IBM.rSto0_StorageRack.nodeBoard[*].node[*].bsModule[*].disk.
       readBandwidth = 650.0Mbps
13 IBM.rSto0_StorageRack.nodeBoard[*].node[*].bsModule[*].disk.
       writeBandwidth = 420.0Mbps
14 IBM_ScenarioA_1server.rSto0_StorageRack.nodeBoard[*].node[*].
       osModule.memory.size = 2.0GiB
```

Listing 4. Data-center configuration in SIMCAN

4 Evaluation

This section presents a experimental study that shows the applicability of our our proposed approach. In order to carry out these experiments, a data-center inspired by IBM Flex system has been modelled (see Figure 2). In this case, the target system contains two racks and one main switch, that is, one computing rack for processing purposes and one storage rack for managing data. The computing rack consists of 6 board nodes, where each board contains 14 p460 blades with 4 CPUs, 64 GB of RAM memory and a local disk drive of 1TB. Hence, the modelled system provides a

total of 336 CPUs. The storage rack has been modelled with one blade consisting of 2 CPUs, 32GB of RAM memory and a high performance disk drive of 2TB. Each rack uses an Ethernet 10/100 network to interconnect the blades. The main switch is connected to each rack through an Ethernet 10-Gigabit network.

This data-center has been modelled using MAGICIAN, and the alternative designs have been simulated using SIMCAN[3], using the code generator explained in Section 3.3. In order to analyze the overall system performance, a Map-Reduce application has been used [5]. This application processes a 2.5GB data-set. This data-set is divided into small data portions, called domains, which are delivered among the different processes. In these experiments, 336 processes are executed in the available CPUs of the system. It is important to remark that each process has a dedicated CPU. The size of each domain is 4MB and the size of generated data, after processing each domain, is 2MB. Each process requires 1,875,000 MIs to process a single domain.

Once the data-center has been modelled, MAGICIAN detects 2 possible inconsistencies in the data-center configuration. The first inconsistency targets the infrastructure of the data-center (the rule detecting this issue is the one in Listing 1), while the second is related to a possible misconfiguration of a single parameter (see Listing 2).

In the first case, the storage rack has been configured to use 1 blade only, that is, 1 storage server. Generally, when the proportion between the number of processes and the storage resources is not properly balanced, the storage system acts as a system bottleneck, slowing down the overall system performance. Consequently, MAGICIAN suggests to increase the number of storage servers by modelling each board in the rack with different storage blades. In order to show the usefulness of this rule, different alternative configurations, using the expert rule described in Listing 1, have been generated. In this case, these simulations have been executed using 1, 2, 4, 8, 16 and 32 storage servers. Figure 5 shows the results obtained from these simulations, where the x-axis represents the number of storage servers and the y-axis represents the speed-up with respect to the initial configuration using 1 storage server. This chart shows that the overall system performance slightly increases when the number of storage servers increases as well. However, this chart also shows that there should be another bottleneck in the system, because the increase is not high.

In the second case, MAGICIAN suggests to change the configuration of the network. Since the original model uses a 10/100 Ethernet network in each rack, it may lead to slowing down the system significantly. Similarly, in this case we have simulated the data-center using a different number of storage servers and a 10-Gigabit Ethernet for communicating the blades in the racks. Figure 6 shows the results obtained from the simulation of the alternative data-center

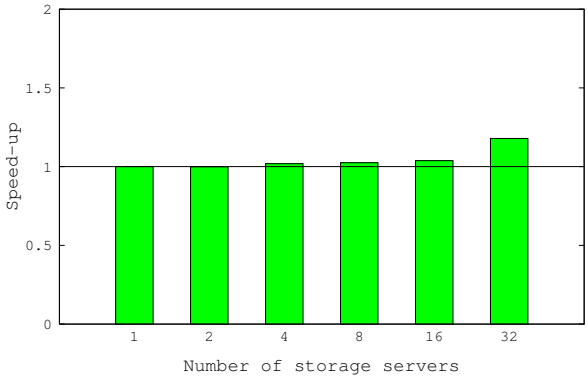


Figure 5. Proposed data-center designs by the expert-rule shown in Listing 1

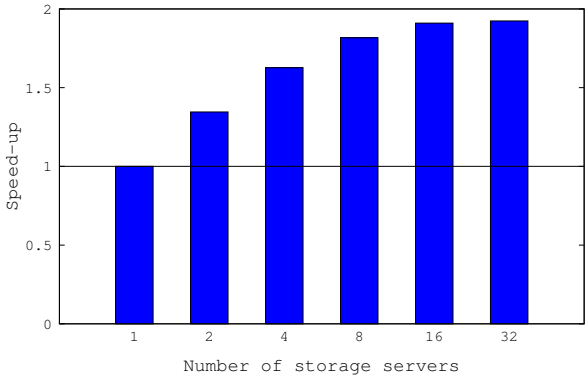


Figure 6. Proposed data-center designs by the expert-rule shown in Listing 2

designs. This chart shows a significant increment of performance when the number of storage servers are increased.

As a conclusion, the initial configuration of the data-center had two drawbacks. First, using only 1 storage server limits the parallelism for accessing data in the system. In this case, MAGICIAN detects an inconsistency in the ratio between the number of CPUs and the number of storage servers. The expert rule suggests to increase the number of storage servers. In particular, this rule recommends to use 8 storage servers for this data-center configuration. Second, the network used in the racks acts as a system bottleneck. In this case, the issue is easily fixed by using the corresponding quick-fix.

5 Related work

The correct design of distributed systems is a process that requires years of expertise. In order to alleviate the inconveniences of this complex and costly task the scientific community has performed a constant effort [6]. Alshahrani and Peyravi presented a theoretical model to design and evaluate communication networks in data-centers [7]. This proposal includes an experimental analysis where the three major DCN architectures have been deployed by using simulation techniques.

In the field of modelling and simulation several contributions can be found. Son et. al presented CloudSimSDN [8], a simulation framework for software-defined cloud infrastructures. This framework incorporates a graphical interface to design the data-center topology. Meisner et. al presented [9], a simulation infrastructure for data-center systems. This approach is based on a higher level of abstraction, and uses a combination of queuing theory and stochastic modeling. Although these works allow to model several infrastructures in a fast and easy way, some of their main weakness are related with the low level of detail of the resultant infrastructures models. In order to alleviate these inconveniences, Nuñez et. al presented SIMCAN [3], a simulation platform designed to analyse and test parallel and distributed architectures and applications.

More recently, Palyart et al. presented MDE4HPC [10], an model-based approach to describe and generate scientific knowledge for diverse architectures. This work presents a methodology to generate HPC applications independently from the platform by using Archi-MDE. Hence, to the best of our knowledge, there is no proposal to design data-center infrastructures that combines expert rules and simulations. Although there exist several simulation platforms, none of them includes users assistance during the modelling process. For this, our approach complements some of the existing simulation platform with expert-rules. In this case, we have selected SIMCAN due to its high level of detail and flexibility. In addition, this SIMCAN simulation platform is based on the OMNeT++, one of the most extended and adopted simulation platforms in the scientific community. Moreover, expert rules are expressed in EVL, which in its turn is based on OCL, a widely used standard for expressing model queries and constraints.

6 Conclusions and Future work

In this work we have presented `MAGICIAN`, a model-based approach for the design and analysis of data-center configurations. The metodology relies on expert rules to detect and fix suboptimal decisions, and on simulation to analyse performance and scalability of the configurations.

We have performed several experiments by modelling a real data-center using `MAGICIAN`. The proposed data-center designs, after applying the suggestion made by `MAGICIAN`, show that the existent inconsistencies in the initial design are fixed. Also, the new designs provide an overall system performance higher than the initial model.

In the future, we would like to support semi-automatic tuning configuration to reach a specific performance goal.

Acknowledgements

Research partially supported by the Spanish MINECO projects DArDOS (TIN2015-65845-C3-1-R) and FLEXOR (TIN2014-52129-R), and the Comunidad de Madrid project SICOMORo-CM (S2013/ICE-3006).

References

[1] S. Tarapore, C. Smullen, and S. Gurumurthi, "Midas: An execution-driven simulator for active storage architectures," in *Workshop on Modeling, Benchmarking, and Simulation*, Beijing, 2008, pp. 1–10.

[2] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*, ser. Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers, 2012.

[3] A. Nuñez, J. Fernández, R. Filgueira, F. García, and J. Carretero, "SIMCAN: A flexible, scalable and expandable simulation platform for modelling and simulating distributed architectures and applications," *Simulation Modelling Practice and Theory*, vol. 20, no. 1, pp. 12–32, 2012.

[4] S. Kolovos, R. Paige, and F. Polack, "Rigorous methods for software construction and analysis." Springer, 2009, ch. On the Evolution of OCL for Capturing Structural Constraints in Modelling Languages, pp. 204–218.

[5] A. Núñez, C. Andrés, and M. G. Merayo, "Optimizing the Trade-offs Between Cost and Performance in Scientific Computing," in *International Conference on Computational Science*, 2012, pp. 498–507.

[6] N. Siegmund, A. Grebhahn, S. Apel, and C. Kästner, "Performance-influence models for highly configurable systems," in *10th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2015, 2015, pp. 284–294.

[7] R. Alshahrani and H. Peyravi, "Modeling and simulation of data center networks," in *Conference on Principles of Advanced Discrete Simulation*. ACM, 2014, pp. 75–82.

[8] J. Son, A. V. D., R. Calheiros, X. Ji, Y. Yoon, and R. Buyya, "Cloudsimsdn: Modeling and simulation of software-defined cloud data centers," in *International Symposium on Cluster, Cloud and Grid Computing*. IEEE Computer Society, 2015, pp. 475–484.

[9] D. Meisner, J. Wu, and R. Wenisch, "Bighouse: A simulation infrastructure for data center systems," in *Int. Symp. Performance Analysis of Systems and Software*. IEEE Comp. Soc., 2012, pp. 35–45.

[10] M. Palyart, D. Lugato, I. Ober, and J. Bruel, "MDE4HPC: an approach for using model-driven engineering in high-performance computing," in *Integrating System and Software Modeling*, vol. 7083. Springer, 2011, pp. 247–261.

Characterizing Relationships for System Dynamics Models Supported by Exploratory Data Analysis

A Conceptualizing Approach about the Meeting Diversity in Student Software Projects

Fabian Kortum, Jil Klünder, Kurt Schneider
Software Engineering Group
Leibniz Universität Hannover
30167 Hannover, Germany
{fabian.kortum, jil.kluender, kurt.schneider}@inf.uni-hannover.de

***Abstract*— Estimating dynamic components in projects involves understanding human factors which are substantial in software development. Communication and collaboration in teams consist of social-driven characteristics with influences on the continuous delivery of software. Efficiently estimated meetings become increasingly important due to budget calculations and shortened release cycles. Experiences of project managers combined with retrospectives on historical data records support a better understanding of team dynamics. But interpreting complex effects is not always trivial, in particular without further analyzes. In several studies, information relationships are investigated through linear correlation measures. Additional analyses for higher correlations are often neglected due to the advanced functional characterization. This leads to statistical gaps with significances for explored data relationships and their functional interpretation. In this paper, we present a systematic identification and visualization of team communication effects and diversities for field study records of 34 student software projects. We combine methodologies from system dynamics with exploratory data analysis to extract and emphasize significant effects. These insights help to sensitize for advanced investigations about the statistical measures of correlation and to interpret sophisticated structures. Furthermore, it reinforces potentials for a team's communication performances and enables an enhanced understanding about how student teams meet and communicate.**

Keywords*— *Team dynamics; exploratory data analysis; data visualization; student software projects

I. INTRODUCTION

Human factors have become increasingly important for software engineering disciplines and processes. Especially the relationships of single factors are often difficult to understand or too complex to be directly interpreted [10]. It remains to the experience and knowledge of project leaders to estimate future projects inter alia according to team structures, capacities, and budget capabilities. The longer a project manager collaborates with a particular team, the more harmonized and predictable becomes the team's typical behavior. Knowledge transfer and exchanges are often achieved by self-reflections and retrospectives of all involved project members at the end of the project [16]. The common use of system records during the development process allows subsequent statistical analysis on available project data, team performances and comparable attributes that can be monitored. This analysis leads to the identification and visualization of important compounds. Illustrating sophisticated information structures and the interpretation of these can be realized in various ways [14]. But interpreting data relationships is often linked with risks if relationships are insufficiently analyzed. This occurs when data dependencies become statistically explained without further proof for multiple kinds of functional characteristics, e.g. determining a data pair's non-linearity. This can cause gaps in interpretation and inadequate significance measures on explored data relationships. However, another problem is the number of records: The more records, the more combinations and types of connections exist. Anyhow, each of these relationships requires further examination for complex functional relationship characteristics [7]. This takes an enormous amount of extra time and effort when analyzing data manually and is almost unrealistic for large data sets.

In this study, we explore some important effects on student teams' communication and meeting behavior. The underlying data set originates from a previous field study focusing on group effects and the communication behavior in 34 student software projects. The data consists of weekly data reports from each team leading to more than 15.000 database entries. This includes communication paths, intensities and network structures, social manner, used media channels, mood and team spirits, meeting quantity, duration and participation from 165 student participants [6]. Our study concerns on meeting diversity effects about the following seven categories of system components shown in Fig. 1.

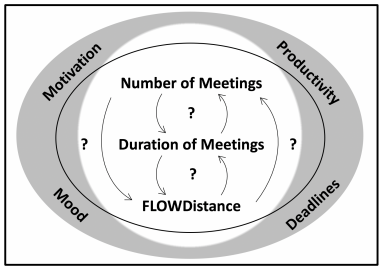


Figure 1. Report components with relevance for team's meeting diversity

DOI reference number: 10.18293/SEKE2017-143

Based on previous studies [6, 8] and a subset of open questions about team communication effects, this approach focuses on the verification of three assumptions with strong relevance to meeting manners during student software projects. Insights about the following questions can help us to optimize the educational concept of student software projects, also identifying stereotypes about students operating manner.

RQ1 is derived from expectations on student's meeting estimation when quality gates occur. In the best case, the meeting times remain at an almost constant level that would represent balanced working process without firefighter situation.

> *RQ1: Does the duration or quantity of team meetings increase for project weeks that are terminated to have quality gates or deadlines?*

Insights from RQ2 present two indicators for the duration of team meetings. For early communication diagnoses and tendency estimators, it is important to understand whether the team's perceived productivity of communications ends up with longer or shorter meeting durations. The same applies for team member's subjectively reported motivation during each project week. A decreasing level of team's motivation can be probably associated with shrinking meeting hours. Thus, it can be one potential indicator for an ongoing motivation condition in teams.

> *RQ2: Does the communication productiveness or team member's motivation have an effect on the duration of team meetings?*

In RQ3, we want to verify whether a decentralized meeting for instance through video chats or other digital channel have positive, negative or even no effects on the motivation of a team. As a matter of fact, face to face communication is an approved contact form with maximum information flow [17]. Compared with other communication channels, face to face communication presents additional perspectives for the motivation in teams.

> *RQ3: FLOW distance is defined as a metric to describe a team's decentralized communication and meetings [10]. Does a decentralized meeting also affect the team's perceived motivation?*

In previous approaches on early communication diagnoses for tendency forecasts, we revealed the importance of characterizing data relationships, that should be made with most maximal aware for accurate estimators [8]. For the verification of RQ1, RQ2, and RQ3, we apply exploratory data analysis to characterize and interpret key components that are later used to describe the meeting dynamics. So far unknown data relationships are analysed through mutual information consideration. At the same time, such techniques resolve a more efficient interpretability of data relationships. These insights can help to sensitize on investigations about the statistical measure of correlation and clarifies the structures.

We also present a simplified process based on techniques for conceptualizing a system dynamics model [1, 12]. The conventional conceptualization of this kind of model is often realized supported by statistical methods, e.g. analysis of regressions, distributions, correlations and data significances. At this moment, semi-automatic statistical methods for identifying linear, non-linear and other functional relationships are rarely taken into account. We introduce the maximal information-based non-parametric exploration (MINE) method which can detect and classify up to 27 different types of functional data relationships [7]. In addition to the verification of RQ1, RQ2, and RQ3, it is also our aim to apply a conceptualizing process for building a system dynamics model with the support of exploratory data analysis on student team records. This allows us to visualize the basic mechanisms of effects within a system. Although, this paper does not include neither the formulation nor the simulation of the system dynamic model. Such models are mostly used to perform experiments and simulations to discover further, not yet identified effects in software projects. However, these subsequent processes will be introduced in later publications and are not part of this paper. Although more research questions are conceivable, we focus on those mentioned above. We will present solutions for visualizing the results, represent the identified communication and meeting effects, give context information about relationships and serve the methodical conception of dynamically influencing components within a system.

II. Origin of Team Communication Records

Software projects at Leibniz Universität Hannover take an important role for students in the fifth semester of their undergraduate computer science studies. These projects fulfill real world's customer requirements, time pressure and self-managing organizations within each developer team. Self-chosen student project leader and quality associates navigate the team through each phase of a waterfall-oriented development process. In an interdisciplinary cooperation with psychologists, Schneider et al. [6] studies both the dynamics of communication behavior and information flow from student software developers since a few years. Several student software projects have been monitored to grasp data about social driven team dynamics and to establish early diagnoses and tendency forecasts for communication diversity [8]. The following list shows conditions about teams and projects from the previously taken field study [6]:

a) 34 student software projects with waterfall-oriented development process and durations of 15 weeks
b) The students were academia undergraduates with at least five semesters in major of computer science.
c) Team sizes: 31 of 34 teams consisted of five people
d) Projects were comparable in complexity, effort of time and fulfilled real end-user requirements.
e) Self-organization: The teams had to manage tasks, project scheduling, social conflicts and problems.
f) Three quality gates were integrated during the project phase to ensure the quality of product.
g) At project's end, all teams had to fulfil a customer acceptance test, individually formed based on each projects requirements compliance.
h) Students have basic knowledge in software engineering disciplines, programming skills in Java, and mixed experiences in working as a group.

III. Related Work

Our research is based on related work with manifested methods in the field of system dynamics modeling as well as the applied statistical information analysis.

Decades ago, Forrester [1] pioneered the well-known system dynamics model, which is a methodology for holistic analysis and model based simulation of complex and dynamic systems. The quantitative modeling represents a strategy to identify and investigate forces within systems which have led to a problem in the past or are of particular importance. The author manifested qualitative models through flow diagrams to simulate system behavior that enables a deeper understanding. Stocks, rates, and auxiliary variables are used to describe system interconnections and show how the effects of action lead to the behavior of systems that are partly non-linear and contra-intuitive [1, 5]. The processes used in this study are based on Forrester's [ref] conceptualization processes to achieve quantitative knowledge about dynamics.

Also, especially for the principles of dynamic events in software project management, Abdel-Hamid et al. [3] introduced various case studies with large software projects providing relevant metrics. They applied data records and experiences to build system models with different stages of complexity. The authors present a detailed overview of plenty dynamic modeling examples like the dependence of productivity on the motivation of development teams and even an entire software development process chain.

Madachy et al. [4] describe the early stages of modeling communication and team issues, including Brooks law. The authors applied qualitative model simulations to understand different process dynamics through regular boundary expressions under a range of parameter settings.

Houghton et al. [2] focused on the data inclusion and consideration for system dynamics modeling procedures. The authors describe investigations about the possibilities for expanding the conventional system dynamics methodology by conceptualizing and formalizing data collections using statistical methods.

IV. Methodology

The methodology of this approach describes the investigative processing of students' communication and meeting diversity in software projects through the novel exploratory data analyzing technique MINE [7]. First, we present MINE's operational advances and power for the identification and characterization even for complex data relationships and structures. We continue with the conceptualization stage for building a system dynamics model that can be later used for visualizing and abstracting mechanism effects within the target system. Parallelly, we apply conventional data visualization techniques that are commonly used to explore linear data relationships like line charts and cross-correlation plots to demonstrate the difficulties and gaps when data dependencies only become expressed as linear relationships. Using the exploratory relationship identification through MINE, we also present force-based network diagram that graphically highlights the affecting and affected team communication components within the system.

A. Data analyses for maximal information coefficients (MIC)

MIC [7] is a novel measure of data dependence that captures linear, non-linear and more complex functional associations between a pair of variables. The algorithm identifies the maximal available mutual information through consideration of relationships types and their functional properties characteristics. This enables the identification and estimation even of a complex association between a pair of data compounds, where conventional statistical correlation measures would solely consider the linearity of relationship. Fig. 2 summarizes MIC's power for the identification of relationships compared to other correlations methods that take common place on data analysis.

Relationship Type	MIC	Pearson	Spearman	Mutual Information (KDE)	Mutual Information (Kraskov)	CorGC (Principal Curve-Based)	Maximal Correlation
Random	0.18	-0.02	-0.02	0.01	0.03	0.19	0.01
Linear	1.00	1.00	1.00	5.03	3.89	1.00	1.00
Cubic	1.00	0.61	0.69	3.09	3.12	0.98	1.00
Exponential	1.00	0.70	1.00	2.09	3.62	0.94	1.00
Sinusoidal (Fourier frequency)	1.00	-0.09	-0.09	0.01	-0.11	0.36	0.64
Categorical	1.00	0.53	0.49	2.22	1.65	1.00	1.00
Periodic/Linear	1.00	0.33	0.31	0.69	0.45	0.49	0.91
Parabolic	1.00	-0.01	-0.01	3.33	3.15	1.00	1.00
Sinusoidal (non-Fourier frequency)	1.00	0.00	0.00	0.01	0.20	0.40	0.80
Sinusoidal (varying frequency)	1.00	-0.11	-0.11	0.02	0.06	0.38	0.76

Figure 2. Power for identifying relationships types using MIC [7]

MIC represents a relationship coefficient score considering several further statistical correlation measures. This includes, for example, the Spearmen correlation coefficient, Kraskov et al.'s [9] mutual information estimation, maximal correlation estimation using ACE [15] and the principle curve-based CorGC dependency measures [11] for the identification of inter-dependencies for up to 27 different functional relationship types. Therefore, the algorithm can automatically determine whether a data relationship is interpretable as linear or more complex functional dependence. Estimating tendency models, these further differentiation results have a significant meaning for the outcome's accuracy of forecasting models. In particular, the algorithm determines the existence of a relationship between two data variables in regression problems and non-linear information analysis. Its analysis is grounded on curve-fitting techniques and resembles an automated function regression with maximal mutual information consideration.

B. Conceptualization of a system dynamics model

The commonly applied building process for system dynamics models is divided into four main stages. Each of these steps consists of sub-steps as shown in Fig. 3. This paper only covers the first stage about the conceptualization of system dynamics models and provide practical advice for completing each sub-step. Due to the scope and limitations of this approach, the three remaining stages in the model building process will take part in continuous publishing. Our aim about a conceptual model is to reach an enhanced understanding of the way how student teams communicate and manage their meetings. We also want to simplify the identification and characterization of effects without loss of quality and gaps due to solely linearity measures.

BUILDING A SYSTEM DYNAMICS MODEL

Stage 1	**Conceptualization (focus of this approach)**
	(1) Define the purpose of the model
	(2) Define the model boundary and identify key variables
	(3) Describe the behavior with reference plots of key variables
	(4) Diagram the basic mechanism of the system
Stage 2	**Formulation**
	(1) Convert Feedback diagrams to functional equations
	(2) Estimate operating ranges through parameter values
Stage 3	**Testing**
	(1) Simulate the model and test the dynamic hypothesis
	(2) Test the models' assumptions
	(3) Test model behavior and sensitivity to perturbations
Stage 4	**Implementation**
	(1) Test the models' response to different policies
	(2) Translate study insights to an accessible form

Figure 3. Forrester's stages for building a system dynamics model [12]

Stage 1.1 Define the purpose of this dynamic system modelling

In the first stage, conceptualizing a system dynamics model helps to understand a model's boundary and operating limitation, influencing factors and effects between components. This encloses the meeting and communication behavior of teams, and also their organizational structures during a waterfall process-driven software project. A model about team communication diversity could be rather used to estimate the tendency course of team dynamics for future projects. The identified relationship characteristics, stereotype, and effects express the model's operating mechanism. Simulation runs and experiments with varying communication constellation additionally can resolve knowledge and information about dynamic behavior over time. Furthermore, explicit situational cases or exploratory project planning and knowledge infusions are of higher interests. Therefore, internal structures and interactions during different phases of the development process are more interpretable and transparent through key components with known relationship effects. The conceptual model in this paper shall be used to abstract and visualize the mechanism of teams meeting diversity according to RQ1, RQ2, and RQ3. Findings of coherent meeting effects become resolved through the exploratory data analyses in MINE.

Stage 1.2 Define the system boundary and identify key variables

The guide about how to build a system dynamics model by Forrester [1] continues with the determination of key variables and components of the target system. All elements within a data set become manually reviewed and marked due to their assumed relevance for the system boundary. The marked elements represent a closed systems boundary within which the behavior, e.g. meeting diversity, is analyzed. The primary components list must be reasonable, aggregated and if possible directionally labeled. For example, the *MoodPositive* element from the initial components boundary list in Fig. 4 represents an aggregated component for ten psychological characteristics like happy or satisfied to express a team's mood. These features are part of the team reports and were documented weekly by each member of a team.

Initial Components Boundary List		
Motivation	Holiday Break	Total No° of Commits
Communication Productiveness	Used Communication Channel	Commitment Rate (Member)
Phase of Project	Change Lines of Code	Team Size
Mood Positive	No° of Meeting Attendants	Missing Member (e.g. Sick)
Mood Negative	Progressed Project Time	Quality Gates
Duration of Meetings	Project Life Span	Requirements Compliance
No° of Meetings	Communication Intensity	FLOWDistance

Dynamic components with feedback mechanism in the system

Endogenous Components Boundary	
Motivation (S)	No° of Meetings (S)
Com. Productiveness (S)	No° of Meeting Attendants (F)
Mood Positive (S)	FLOWDistance (S)
Mood Negative (S)	Communication Intensity (S)
Duration of Meetings (S)	Communication Channel (F)

Components without direct affects by the system

Exogenous Components Boundary	
Phase of Project	Holiday Break
Missing Member (e.g. Sick)	Progressed Project Time
Quality Gates	Project Life Span
Team Size	

Figure 4. System boundary with endo- and exogenous component separation

The review of all data variables, especially the subjectively selected system boundary components requires a subsequent categorization. All components in the initial list need to be classified and separated into endogenous or exogenous system elements. However, Fig. 4 only presents an orientation guideline as summarized overview of possible system components and does not necessarily bind the model's frame. After further examination, the initial boundary list also contains system components that seem to be unnecessary for the current research questions. The components with assumed relevance for the ongoing investigations about the three research questions should be classified as an endogenous or exogenous component to set borderlines for this approaches system boundary. Exploratory data analysis like MINE [7] also helps to identify key elements of a sophisticated system as well as analyzing the maximal available mutual information between associations. The key factor findings in MINE plotted as a weighted network diagram are shown in Fig. 5. Beside the *ProgressedTimeOfProject*, the components *FLOW Distance*, *ProjectPhase, NumberOfTeam-Meetings* and *NumberOfAttendedMeetings* seem to be of central relevance for the system.

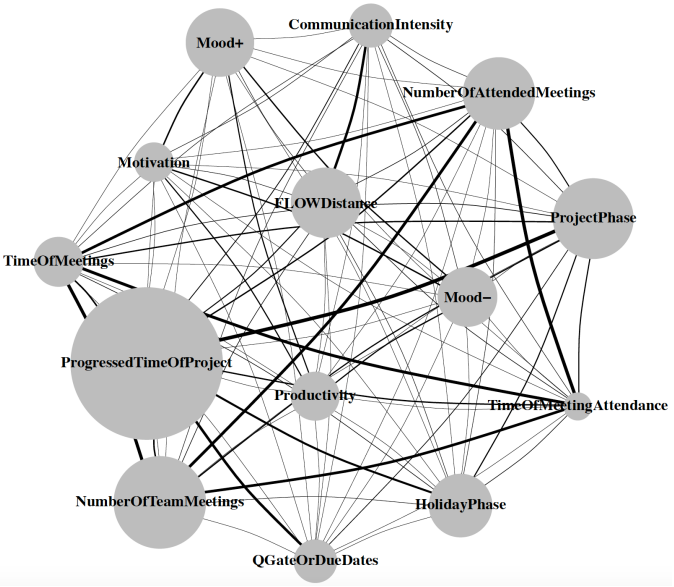


Figure 5. MIC-weighted network diagram on team meeting components

This network plot enables an overview of all investigated component's relationship with statistically identified maximal information coefficient strengths. The diagram nodes present varying sizes according to each measured ongoing MIC strength.

Nodes or component relationships are intersected through edges that vary in their intensity to express the relationship strengths for identified functional properties through MINE. The graphics are realized through a minimalistic Java tool reading data records from CSV-files, performing exploratory data analysis through integration of the MINE application and plotting the identified relationship as an MIC-weighted network diagram. For plotting the graphic, the tool uses an embeddable R-library named igraph [14]. This library also allows extended network visualizations as shown in Fig. 8. It can derive force-based network diagrams with positive and negative notated arrows to describe an effects polarity.

Stage 1.3 Describe the key variables behavior as reference plot

The traditional conceptualization process for system dynamics models by Forrester [5, 12] continues with plotting the assumed components of interest. Through cognitive reviews and interpretations, the difficulty remains in identifying distinct phenomena, patterns or other relationship types between the components. The line chart in Fig. 6 shows meeting characteristics of the teams and their dynamic changes during a project.

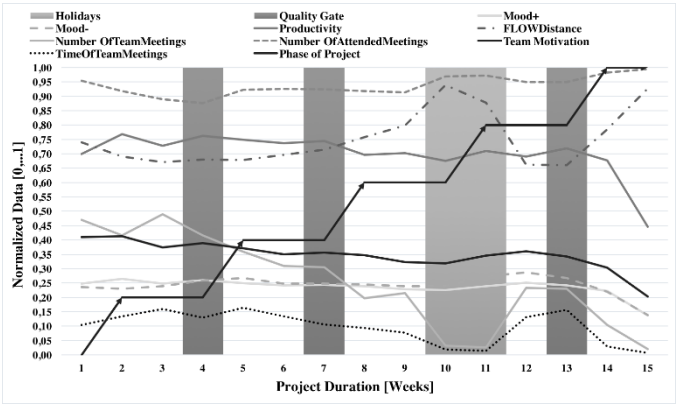


Figure 6. Reference chart for comparisons of team meeting dependencies

The more component's and their changes over time became plot and compared with other elements, the more difficult it is to make qualitative interpretations for data relationships and types. Cross-correlation plots in R-statistics help to measure two component's linear relationship statistically as in Fig 7. The applied visualizations foster a general understanding for meeting diversities and data variances over time. However, it represents an insufficient method, especially for detailed identifications of component's relationships in larger or complex systems.

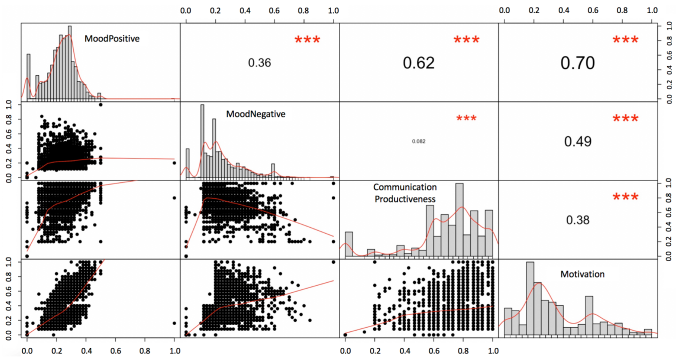


Figure 7. Linear data analysis through cross correlation matrix in R-statistics

As an example, MINE identifies relationship strengths for the components *MoodNegative* and *CommunicationProductiveness* in Fig. 7 differently compared to the linearity measures from the cross correlations in R. In this qualitative comparison, MINE measures a stronger correlation coefficient with strong significance based on the additionally considered functional relationship properties. The example demonstrates the statistical outcome with identification gaps in case of sole linearity analyzes without further relationship characterization. Both correlation coefficient measures and their particular statistical significance with Fisher's exact tests [13] are shown in Tab 1.

Table 1. Identification of relationship strength: MIC vs. Pearson correlation

Component X	Component Y	MIC	sig. *p*	Pearson Cor.	sig. *p*
Mood-	Productiveness	0.30527	0.00780	0.08199	0.0004

For the verification of RQ1, RQ2, and RQ3, we consider all team records for the exploratory data analysis in MINE. This step enables to identify each possible relationship within the data components, shown in the network diagram in Fig. 5. As the subsequent step, we limit the considered boundary of relationships from the entire data structure to only relevant subjects for the RQs. This centralizes the focus and simplifies the interpretability for particular component's effects. In particular, it enables to compute only the sub-network diagram that describes a component's direct interplay as an affected or affecting unit. The Java tool that was established to analyze the team data records includes a feature to select components for managing the focus of relationship analyses in MINE. Therefore, resulting network diagrams can shrink and expand their structure, depending on the selected items and research focus. According to RQ1, RQ2, and RQ3, we chose all components that were identified through MINE to be affected by elements or affect others. On the affected side, we mark the features that were of interest to the research questions. This includes five components: *DurationOfTeamMeetings, FLOW-Distance, NumberOfTeam-Meetings, Motivation*, and *Productivity* (communication productiveness). The force-based sub-network diagram in Fig. 8 shows all affected components with interest for the RQs, as well as all the elements that are affecting these.

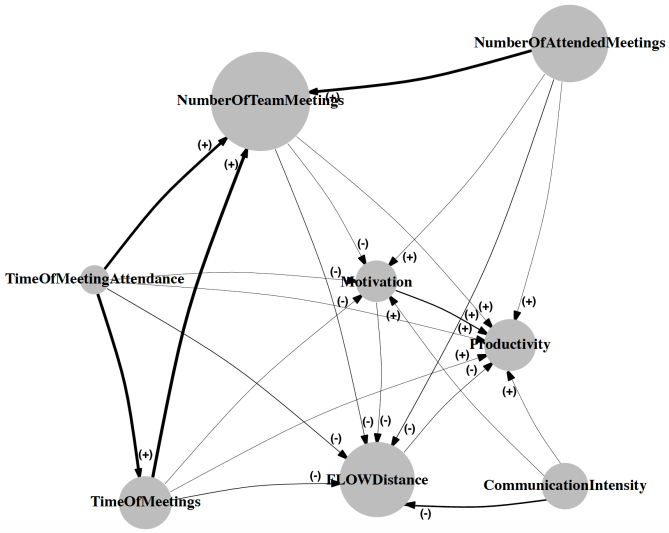


Figure 8. Relationships with directed polarities between meeting components

This diagram is a visualized result of the exploratory relationship identifications through MINE and our Java tool which performs the embedded visualization steps. In Tab. 2, all findings through MINE are listed numerically including their references to the conventional correlation measures on a relationship's linearity.

Table 2. Relationship identification and MIC-scoring through MINE

Effecting Component X (identified with MINE)	Affected Component Y (selected scope)	MIC (strength)	MIC-pearson² (nonlinearity)	MEV (functionality)	Linear regression (pearson)
TimeOfMeetings	NumberOfTeamMeetings	0,88203	0,37842423	0,88203	0,7096519
TimeOfMeetingAttendance	TimeOfMeetings	0,81826	-0,040376246	0,81826	0,92662627
NumberOfAttendedMeetings	NumberOfTeamMeetings	0,79197	-0,077647746	0,76115	0,93253297
TimeOfMeetingAttendance	NumberOfTeamMeetings	0,73744	0,28851897	0,73744	0,6700157
CommunicationIntensity	FLOWDistance	0,68798	-0,05492699	0,68668	-0,86192054
Motivation	Productivity	0,30437	0,16183046	0,30437	0,37754408
TimeOfMeetingAttendance	FLOWDistance	0,19455	0,054150358	0,19455	-0,37469938
TimeOfMeetings	FLOWDistance	0,19312	0,054751784	0,19312	-0,3719788
NumberOfAttendedMeetings	FLOWDistance	0,18831	0,03741516	0,18831	-0,38845184
NumberOfTeamMeetings	FLOWDistance	0,1804	0,024076387	0,1804	-0,3953778
CommunicationIntensity	Motivation	0,14371	0,075237975	0,14371	0,2616716
Motivation	FLOWDistance	0,1404	0,097615935	0,1404	-0,20684311
CommunicationIntensity	Productivity	0,13312	0,015243918	0,13312	0,34333086
FLOWDistance	Productivity	0,1305	0,05606755	0,1305	-0,27282313
TimeOfMeetings	Motivation	0,08316	0,07907622	0,08316	-0,06390443
NumberOfTeamMeetings	Motivation	0,06959	0,06958476	0,06959	-0,002290251
TimeOfMeetingAttendance	Motivation	0,06919	0,06621243	0,06891	-0,05456712
NumberOfAttendedMeetings	Motivation	0,06334	0,06332517	0,06334	0,003851846
TimeOfMeetings	Productivity	0,05778	0,03728396	0,05778	0,14316438
NumberOfTeamMeetings	Productivity	0,05536	0,03248602	0,05536	0,15124145
TimeOfMeetingAttendance	Productivity	0,05209	0,03416415	0,05209	0,13388745
NumberOfAttendedMeetings	Productivity	0,04873	0,03124734	0,04873	0,13222201

The used Java tool enables to mark whether an effect has negative or positive influences as directed polarity label for each identified relationship. Each polarity is detected through the trendline course of the identified functional relationship between a pair of components. The formerly visualized relationships in Fig. 8 with the directed polarities combined with the statistical measures in Tab. 2 allow us to interpret and describe the behavior of each component and therefore to verify the RQs.

According to RQ1, we want to prove whether quality gates are a reliable indicator for an increasing number or duration of team meetings. The methodology of this study allows to verify the RQs in two ways: Through statistical measures and also as cognitive interpretation for visualized relationship networks. The secondary is less accurate but has strong benefits for simple relationship representations. The affected component *NumberOfMeetings* in Fig. 8 reveals three incoming positive influences from the affecting components *NumberOfAttendedMeetings*, *TimeOfMeetings*, and *TimeOfMeetingAttendance*. There is no evidence in the empirical records confirming the dependency between an increasing number of team meetings and occurring quality gates. This is also correct for the duration of team meetings. The component *TimeOfMeetings* is only affected by the time of students' typical attendance in a meeting. Therefore, we cannot confirm that *QualityGates* influence the quantity and duration of student team meetings.

RQ2 is about whether the communication productivity or motivation have an influence on the duration of team meetings in the student projects. We show in Fig. 8, that the component *TimeOfMeetings* is affected by the component *TimeOfMeeting-Attendance*. It also affects four others differently i.e. positively the *FLOWDistance*, negatively the *Motivation*, positively *Productivity* and positively the *NumberOfTeamMeetings*. All these components have endogenous characteristics within the system. Means they all represent affected elements, but also can affect other components at the same time. Vice versa, the diagram indirectly obtains that *TimeOfMeetings* will be reduced with an increasing *Motivation*. An increasing communication *Productivity* leads to a positive effect on the *TimeOfMeetings*. Therefore, the higher the perceived communication productivity of team members, the more time they will spend in meetings.

RQ3 concerns potential effects on a team's perceived motivation in the case of distributed communication structures which can be expressed by a high *FLOW-Distance*. Fig. 8 that this component is negatively affected by the *Motivation*. Both components are endogenous units as well. Therefore, they also consist of feedback loop mechanisms. For instance, this means that an increasing *FLOWDistance* will decrease the team's perceived *Motivation* within a project and vice versa. The graph also shows that a decentralized communication has plenty effects on e.g. quantity, duration, participation in meetings with relevance for teamwork.

Stage 1.4 Diagram a system's basic mechanism

The dependencies in our system with specific component scopes can be expressed through the notated polarity interactions shown in Fig. 8. So far, these diagrams do not cover or describe feedback loops for components' interplay. Forrester [5] applies quantitative modeling through stocks and flows diagrams as in Fig. 9 to investigate interactive effects on endogenous components. The stocks represent components controlled through auxiliaries. Each auxiliaries function describes a single stock's level change over time by associated effects that were identified to influence a stock. In fact, auxiliaries regulate effects through functional equations, which is part of the second stage to formulate a system dynamics models. However, this paper limits its focus on the conceptualization of a system dynamics model. We manually build the stock and flow diagram in Fig. 9 to quantitatively visualize the dynamic mechanism regarding the RQs about students' meeting diversity in software projects. The model is realized with the educational license for the modeling and simulation software Anylogic.

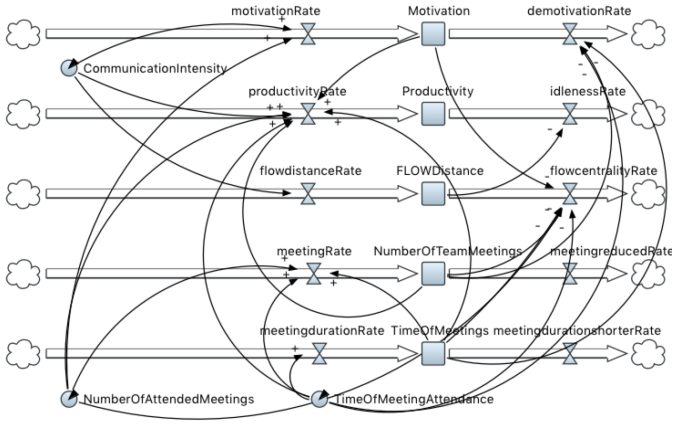


Figure 9. Stock and flow diagram for a meeting diversity system

This quantitative model abstracts the component's interplay through Forrester's notation REF. Formalizing the so far only conceptual auxiliaries will resolve a qualitative model that also enables simulations. However, this requires manually effort by analysts working on the *Formulation* stages listed in Fig. 3. Formulating relationships also requires detailed knowledge about the type of connection. It is an incremental building process that helps project analysts to detect further heuristics and gather information for all project associates.

V. Interpretation and Validity of Results

The objective of this research was to combine the system dynamics terminology with exploratory analysis on information provided by student reports during software projects. The MINE methodology and visualization techniques, e.g. the conceptual modeling and force-based networks are applicable for further system conceptualization and data structure characterization [7].

A. Interpretability of statistical results

Before we started with our investigations, we approved the quality of data records. Therefore, we performed univariate variance analysis for all data records through ANOVA to ensure that the data is normally distributed. For the comparison of different data types, all analyses are done with normalized values. The applied RQs are defined due to assumptions with relevance on student teams' meeting behavior in university software projects. All relationships listed in Tab. 2 are verified through Fisher's exact tests with p-values < 0.05, i.e. with significances. The explored insights help to enhance framework conditions and educational concepts for future student software projects. Supplementary, the identified meeting dependencies in Fig. 8 characterizes that a centralized communication structure has strong importance for the motivation and positive atmosphere in teams. Beside newly gathered insights, some trivial relationships could be noted as well like that a team's motivation has positive effects on the communication productivity in meetings. Additional analyses on other teams and software projects probably discover more aspects of dynamic team behavior and show human factors in a more understandable way.

B. Threats to validity

This approach underlies several threats to validity. All results are derived from the student records in a previously taken field study [11]. Experience records of other teams with different framework conditions may lead to other results. The authors subjectively set the initial system's component boundary listed in Fig. 4 with assumed relevance for meeting factors. Different researcher's experiences may result in other selections. Due to the nature of system conceptualization, our meeting model might be incomplete and missing exogenous influences. The reliability and quality of insights about the meeting diversity in teams are only statistically validated and resolutely remain solely applicable for other student software projects with similar frameworks. Consequently, the results should not be over-generalized for regular developer teams in software projects.

VI. Conclusion

We introduced the conceptualization steps about modeling system dynamics [12] in combination with the MINE terminology [7] that establishes an advanced identification of functional relationships in sophisticated data structures. The objective of this study is to resolve a better understanding about dynamic dependencies on meeting diversity in student software projects. Human behaviors are not always trivial to comprehend. Therefore, we applied field study records from 34 student software projects to an exploratory analysis and visualization proceeding, which verified three research questions about meeting behavior with assumed relevance for educational perspectives on project scheduling. Beside the system dynamics mechanism, we derived force-based network diagrams providing a simplified visualization for meeting dependency structures that are also understandable not especially for analysts. We could prove that the centrality of meetings takes an important role in the communication productiveness and motivation of student developer teams.

The exploratory terminology helps us to characterize the meeting dependencies through functional property analyses on data relationships which are verified as an enhanced identification to complement gaps that are not covered by solely linear correlation measures [7]. Data collections from completed software projects are valuable goods for future decision and planning improvements. It is desirable to simplify the interpretability of structural dependencies information. This study helps researchers and educational staff to understand better and interpret student's meeting diversities, also how to statistically analyze and visually diagram effects in teams.

Acknowledgments

This work was funded by the German Research Foundation under grant number 263807701 (TeamFLOW, 2015-2017).

References

[1] J. W. Forrester, *World dynamics*. Wright-Allen Press. 1971.

[2] J. P. Houghton, M. D. Siegel, A. Wirsch, A. Moulton, S. E. Madnick, & D. K. Goldsmith, A Survey of Methods for Data Inclusion in System Dynamics Models. MIT. Engineering Systems Division. 2014.

[3] T. Abdel-Hamid, & S. E. Madnick, *Software project dynamics: an integrated approach*. Prentice-Hall, Inc. 1991.

[4] R. J. Madachy, *Software process dynamics*. John Wiley & Sons. 2007.

[5] J. W. Forrester, System dynamics, systems thinking, and soft OR. *System dynamics review*, *10*(2-3), 245-256. 1994.

[6] K. Schneider, O. Liskin, H. Paulsen, & S. Kauffeld, Media, mood, and meetings: related to project success? *ACM Transactions on Computing Education (TOCE)*, *15*(4), 21. 2015.

[7] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, ... & P. C. Sabeti, Detecting novel associations in large data sets. *Science*, *334*(6062), 1518-1524. 2011.

[8] F. Kortum, & J. Klünder, Early Diagnostics on Team Communication: Experience-Based Forecasts on Student Software Projects. *Quality of Information and Communications Technology (Quatic)*. IEEE. 2016.

[9] A. Kraskov, H. Stögbauer, & P. Grassberger, Estimating mutual information. *Physical review E*, *69*(6), 066138. 2004.

[10] J. Klünder, K. Schneider, F. Kortum, J. Straube, L. Handke, & S. Kauffeld, Communication in teams-an expression of social conflicts. *International Conference on Human-Centered Software Engineering* (pp. 111-129). Springer International Publishing. 2016.

[11] P. Delicado, Another look at principal curves and surfaces. *Journal of Multivariate Analysis*, *77*(1), 84-116. 2001.

[12] J. W. Forrester, & S. Albin, Building a System Dynamics Model Part 1: Conceptualization. Massachusetts Institute of Technology. 1997

[13] R. A. Fisher, Tests of significance in harmonic analysis. Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character, 125(796), 54-59. 1929.

[14] K. Ognyanova, & P. Monge, A multitheoretical, multilevel, multidimensional network model of the media system: Production, content, and audiences. *Annals of the International Communication Association*, *37*(1), 67-9. 2013.

[15] L. Breiman, & J. H. Friedman, Estimating optimal transformations for multiple regression and correlation. *Journal of the American statistical Association*, *80*(391), 580-598. 1985.

[16] A.Birk, T. Dingsoyr, & , T. Stalhane, Postmortem: Never leave a project without it. *IEEE software*, *19*(3), 43. 2002.

[17] S. W. Ambler, Agile modeling. 2002

A Software Framework for Data Provenance

Tassio Sirqueira[1], Marx Viana[1], Nathalia Nascimento[1] and Carlos Lucena[1]
[1]Pontifical Catholic University of Rio de Janeiro
Software Engineering Laboratory (LES)
Rio de Janeiro, RJ – Brazil
{tmartins, mleles, nnascimento, lucena}@inf.puc-rio.br

***Abstract*— Data provenance refers to the historical record of the derivation of the data, allowing the reproduction of experiments, interpretation of results and identification of problems through the analysis of the processes that originated the data. Data provenance contributes to the evaluation of experiments. This paper presents a framework for data provenance using the W3C provenance data model, called PROV-DM. Such framework aims at contributing to, and facilitating, the collection, storage and retrieval of provenance data through a modeling and storage layer based on PROV-DM, yet is compatible with other representations of PROV such as PROV-O. To demonstrate the utilization of the framework, it was used in an IoT application that performs the gas classification to identify diseases.**

Keywords- Data Provenance; PROV Framework; PROV W3C.

I. INTRODUCTION

The term "provenance" refers to the origin or provenance of data, that is, it is a record of the data derivation history, which enables reproducibility of experiments, interpretation of results and diagnosis of problems [1]. The provenance is the complementary documentation of a data, containing information of "how," "when," "where" and "why" the data was obtained, "who" got it and "how much" cost this in time or effort.

The provenance provides a look beyond the specifications of domains and suggests the adoption of disciplined models, where data provenance information can be used to learn or understand design methods and rules. It can further assist users in similar investigations in order to understand data correlations and to improve future investigations. Currently, the provenance of data is successfully applied in different areas, mainly e-science [2].

One of the problems of provenance refers to the lack of agreement as to the comprehensiveness of the data to be captured, in addition to the absence of a clear definition of how this procedure should be carried out [7]. Other issues raised with respect to the provenance of data are reliability of data, integrity, confidentiality about its use, availability to other people, beyond efficacy in relation to what is being captured and the efficiency with which this is done, ensuring that all relevant information is captured.

The goal of this section is to discuss the benefits that the data provenance can bring, considering the captured data, how they were modeled and stored, and the type of information to be obtained from them. In this context, we present a framework for data provenance (FProvW3C).

DOI reference number: 10.18293/SEKE2017-085

To illustrate the use of FProvW3C, we will present an example from a system based on the Internet of Things (IoT) [13]: a system that collects data from gas sensors and assists in the classification of gases emitted by humans. We have chosen this example because IoT is an exciting and emerging approach that has gained both academic and industrial attention.

The remainder of this paper is organized as follows. Section II presents the W3C PROV [3] model. Section III discusses related work. Section IV details the FProvW3C and Section V presents the framework instance. Finally, Section VI presents our conclusions and future work.

II. PROV W3C

According to [4], the data provenance can be divided into three types: i) Prospective: it is the sequence of processes used in data generation; ii) Retrospective: this is the information obtained during the execution of data and environment generation processes, and iii) User data: any information that the user deems necessary for future analysis. In addition, the provenance can be obtained in two ways according to [8], which are: i) Lazy: the provenance is obtained from the moment that its capture is requested, and ii) Anxious: provenance is obtained at all times and is readily available. The best way to collect it will depend on the application to be used.

Currently, there are two main patterns for data capture from provenance: i) the OPM model [5], with three vertices, five causal relationships, and ii) the PROV model [3], with three main vertices and seven basic relations, plus complementary ones. In this work, the provenance model used was the PROV [3] by its amplitude and greater number of causal relations for knowledge representation.

The PROV [3] model consists of 12 documents that define their specification. Among the main documents are the PROV-DM, which specifies the data capture model; the PROV-CONSTRAINTS, which is the set of constraints applicable to the data model (PROV-DM), and the PROV-O, an ontology for mapping the data model.

The PROV-DM creates a separation of types and causal relations, the types being: i) Entity: it is either a physical, digital or conceptual type, or something with fixed aspects, in that entities can be real or imaginary. ii) Activity: it is something that occurs over a period time and acts on entities. iii) Agent: it is something that has some kind of responsibility for the activity and the existence of an entity, or for the activity of another agent. Agent, in the PROV model, can be classified as an organization, a person, or a software agent.

In relation to causal relations they are divided into two subsets: primary and secondary (optional) relations. Fig. 1 shows the primary relations in bold and Fig. 2 presents the (optional) secondary relations.

		Object		
		Entity	Activity	Agent
Subject	Entity	**WasDerivedFrom** Revision Quotation PrimarySource AlternateOf SpecializationOf HadMember	**WasGeneratedBy** WasInvalidatedBy R T L	**WasAttributedTo**
	Activity	**Used** WasStartedBy WasEndedBy R T L	**WasInformedBy**	**WasAssociatedWith** R
	Agent	—	—	**ActedOnBehalfOf**

Figure 1. Primary relations of the PROV[1].

		Secondary Object		
		Entity	Activity	Agent
Subject	Entity	—	WasDerivedFrom (activity)	—
	Activity	WasAssociatedWith (plan)	WasStartedBy (starter) WasEndedBy (ender)	—
	Agent	—	ActedOnBehalfOf (activity)	—

Figure 2. Secondary relations of the PROV[2].

One of the advantages of using the PROV and the PRV-O ontology is that PROV-DM can be represented by using OWL2 (Web Ontology Language) [6]. They can also be used to represent and exchange information of provenance generated in different systems and in different contexts.

In addition, another advantage of using the PROV model is with respect to the storage model, where different sources of information are converted into a model standardized by the W3C. This in turn facilitates the understanding, traceability and reproducibility of a data, due to the process that originated it. Furthermore, it allows semantic annotation using the PROV-O.

The provenance may be an important quality metric in the experiment, since the data derivation process has implications for both data quality and the errors introduced by faulty data as they propagate in other derivations [9]. Provenance in the experimentation process can help increase the validity of the experiments, since the reliability of the data will be monitored.

The main objective of the FProvW3C framework is to simplify the capture of provenance data and to facilitate the use of the PROV model, thus allowing for more reliable experiments.

III. Related Work

Provenance can be used as a quality metric for data evaluation because it has significant implications on data quality and errors introduced by faulty data, which increase as derivations are propagated [9]. To support the research developed in this article, a structured search was carried out. Among the results were selected studies that are applied explicitly to the data provenance based on the PROV model.

Starting with ProvToolbox[3] is a Java library to create PROV-DM representations and convert them between RDF, PROV-XML, PROV-N, and PROV-JSON. It is not geared towards capturing and storing data in a DBMS (Database Management System), moreover, is a set of independent tools for each form of representation of the data.

The prov-api[4] is a Java API to create and manipulate provenance graphs. Currently, API only implements PROV's essential terms. The focus of the prov-api is the inference and query in the PROV-O ontology and not in the data storage using the PROV-DM, as well as its use in data capture from provenance.

The PROV Python library[5] is a library that provides an implementation of PROV-DM in Python. Although it is a library close to the concerns of the framework of this article, being in python makes it difficult to capture data coming from multi-agent systems, since most agent platforms are in Java.

The E-SECO ProVersion presented by [2], is a management platform for scientific workflows. Although the application works with provenance data in the PROV-DM model, it uses a lazy approach to data capture. In addition, the PROV model is integrated to the platform code, making it difficult to reuse. The ProvManager, presented by [7], is a data storage and analysis tool, uses prolog for queries and, like E-SECO, does not have mechanisms for integration with other systems, depending on a particular form of data entry.

For his part [10], presents what he later called Prov-Process, a platform for collection, storage and analysis of provenance data. However, a standard model must be used for data entry in the ".csv" format and does not allow integration with other systems.

Although there are several applications aimed at the provenance of data, they do not have features that assist users in data capture and storage. In the next section the details of the framework will be presented.

IV. FProvW3C – A Framework For Data Provenance

As addressed by [2], the data provenance is something constant, and it should follow all the steps performed to obtain concise results. One way to observe this is to consider the life cycle of a data, where not only the data is important but also the process that originated it.

The FProvW3C Framework works with an anxious approach [8], so that the data is collected at all times and can be consulted next. Currently, in its architecture the FProvW3C framework presents all the specification of the PROV-DM with the annotations in Java Persistence API (JPA), according to Fig. 3.

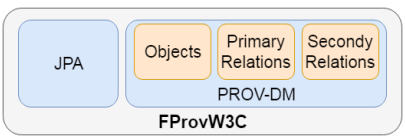


Figure 4. Framework architecture.

[1, 3] PROV-DM: http://www.w3.org/TR/prov-dm/

[3] http://lucmoreau.github.io/ProvToolbox/

[4] https://github.com/dcorsar/prov-api/

[5] http://pypi.python.org/pypi/prov

Besides modeling, the FProvW3C framework brings all persistence annotations. This reduces the work of users and avoids mapping errors by those who do not have extensive knowledge of the PROV. Being a framework makes it possible to integrate it into different systems. The framework frozen-spots are the main vertices and the hot-spots are the causal relations, both defined by the PROV model, and they adapt to different application contexts. The framework was developed in Java and the annotations are based on Java Persistence API (JPA), which makes the framework independent of the DBMS that will be applied to store the collected data that leaves this choice up to the user.

The FProvW3C is relational object mapping framework in charge of creating the database, the tables and their respective attributes in the DBMS. The PROV model determines the classes and how they relate in addition to the basic attributes. In this way, the framework provides the classes with the basic attributes (frozen-spots) and, moreover, allows the creation of hot-spots by extending both the attributes of each class and the relationships. These attributes are intended to represent the characteristics of the system in which FProvW3C will be applied. As the framework performs data mapping according to the PROV, its use is made easier for the user, where are able to apply the data provenance in their applications.

V. Usage Scenario: Gas Analysis

As described by [11], a person emits various gases from different parts of the body (e.g. flatulence, eructation, exhalation) and these gases could be useful for the diagnosis of a set of intestinal and stomach diseases.

Nevertheless, there are very few technology approaches to facilitate the analysis of flatulence and other gases daily emitted by humans. In addition, there is a need for high data reliability in these approaches, since it uses the identification of gas-based diseases, i.e., all information must be reliable and agents cannot fail to capture or classify data.

The FProvW3C framework was used in the "Gases Device" application [11]. This application is based on the Internet of Things (IoT), which uses sensors to measure gases in the environment and uses software agents to provide data classification. Fig. 7 shows the Gases Device application architecture extending FProvW3C classes.

A. Overview

FProvW3C creates an intermediary layer between the application and the database. This layer makes it possible to treat and map the data to be stored by linking the information source to them. In addition, FProvW3C creates a knowledge base based on the use of the application. Therefore, all data entered by users or sensors into the application, as well as data manipulated by the application are registered in this base.

Fig. 6 shows that every time a gas sensor captures the variation of a gas, the persistence of the data in the database is invoked via the FProvW3C framework, registering data provenance. In this way, all the capture and manipulation of data are recorded, such as a filming, forming the historical basis of the application.

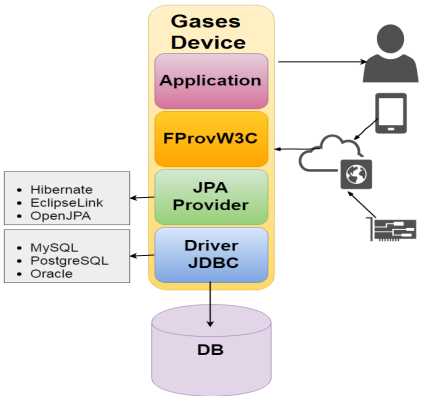


Figure 6. Gases Device architecture with FProvW3C.

B. Results and Discussion

The recorded data of this application includes information from system users, sensors, monitored gases and software agents [11]. The recorded data were linked to the actions of users and software agents, helping in the traceability of each executed action. For example, Fig. 7 illustrates the registration moment of a user with obesity in the Smell App Website. To elaborate an initial database to improve the system's classification, first users were asked to inform personal health information before using the Gases Device. As shown in Fig. 8, the system attributed the ID 36 to this registered user. Then, by using the FProvW3C's structure, it was possible to track all data generated during this registration operation. We can verify in Fig. 8 that the person agent with User ID 36 was successfully created, and the entity "Obesity" was attributed to this agent.

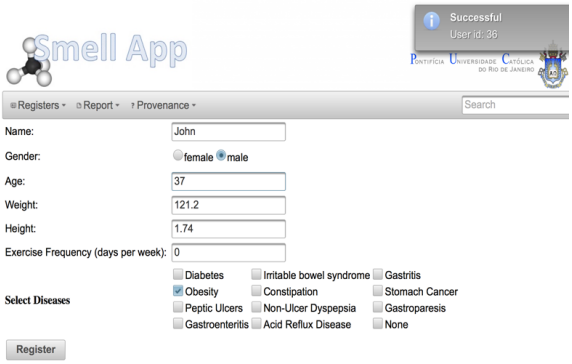


Figure 7. Smell App Screenshot.

After the user had registered, he/she connected a Gases Device to the system and started an exhalation report. This action initiated three software agents: i) Gases Device Agent, which collected data from Arduino; ii) Analyzer Agent, which preprocessed and saved data on the database, and iii) Alert Agent, which evaluated all exhalation reports based on the diseases described in [11] and generated alerts.

To evaluate the operation of a multi-agent system is not a trivial task [12]. As data come from several sources, it is difficult to identify the origin of a problem. However, as shown in Fig. 9, the provenance facilitated the evaluation of this multi-agent system by allowing us to track all activities that were performed during its execution.

PROVENANCE DATA

AGENT

ID Agent	ID User	Type
5	35	Person
6	36	Person

ENTITY

ID Entity	Name
10	Diabetes
11	Obesity

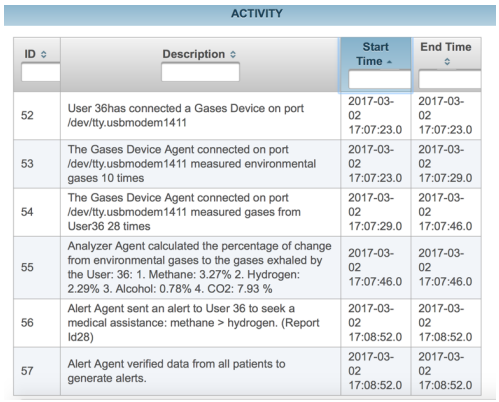
Was Attributed To

ID	ID Agent	ID Entity
3	5	10
4	6	11

ACTIVITY

Was Associated With

Figure 8. Provenance of Smell App Data.



ACTIVITY

ID	Description	Start Time	End Time
52	User 36has connected a Gases Device on port /dev/tty.usbmodem1411	2017-03-02 17:07:23.0	2017-03-02 17:07:23.0
53	The Gases Device Agent connected on port /dev/tty.usbmodem1411 measured environmental gases 10 times	2017-03-02 17:07:23.0	2017-03-02 17:07:29.0
54	The Gases Device Agent connected on port /dev/tty.usbmodem1411 measured gases from User36 28 times	2017-03-02 17:07:29.0	2017-03-02 17:07:46.0
55	Analyzer Agent calculated the percentage of change from environmental gases to the gases exhaled by the User: 36: 1. Methane: 3.27% 2. Hydrogen: 2.29% 3. Alcohol: 0.78% 4. CO2: 7.93 %	2017-03-02 17:07:46.0	2017-03-02 17:07:46.0
56	Alert Agent sent an alert to User 36 to seek a medical assistance: methane > hydrogen. (Report Id28)	2017-03-02 17:08:52.0	2017-03-02 17:08:52.0
57	Alert Agent verified data from all patients to generate alerts.	2017-03-02 17:08:52.0	2017-03-02 17:08:52.0

Figure 9. Provenance of Activities of Software Agents and Users.

This section shows that data provenance can be used for a wide range of purposes in computational applications. Furthermore, when we tracked the system execution in order to understand its operation, we also used data provenance to verify errors in data classified by agents and in data collected from sensors.

VI. Conclusion and Future Work

Registering the data provenance is a necessity in several scenarios, especially those that have complex execution. It is necessary to have a history of each of the steps. The PROV model aims at storing data provenance in a detailed manner, focusing on the responsibilities of agents in each item of provenance.

This paper proposes the FProvW3C framework, designed to capture and store data provenance using the PROV model of W3C. The data provenance helps to trace the origin of the data and the derivation processes that occurred between the origin of the data and the state in which the data is currently found. Considering that the provenance model contributes to evaluate the quality of the data and consequently the process that generated it, this helps increase the validity of the experiments since the data is monitored. The data provenance could be used in the construction of knowledge bases that help in the: i) traceability of actions; ii) identification of errors; iii) follow-up of the steps of a study, and iv) viability of verify results.

For future work, we hope to expand the FProvW3C framework so that it can convert the captured data to an ontology following the PROV-O model. For example, it is possible to extend FProvW3C to support: i) data semantics and syntax, and ii) the ontology of PROV. In addition, we aim at exploring the data provenance in multi-agent systems, recording each piece of information about the agent, its relations with other agents and with the external environment. As such, we could capture information from the agent's decision-making process, expanding the work of [11] and we could use the information obtained to help track errors and answer questions about the agent's behavior, since it is an autonomous entity.

Acknowledgment

The authors thank CAPES, Faperj, CNPq, PUC-Rio and LES for the support, financial resources and encouragement to the research.

References

[1] Lim, C., Lu, S., Chebotko, A., & Fotouhi, F. (2010). Prospective and retrospective provenance collection in scientific workflow environments. In 2010 IEEE International Conference on Services Computing. p. 449-456. IEEE.

[2] Sirqueira, T. F., Dalpra, H. L., Braga, R., Araújo, M. A. P., David, J. M. N., & Campos, F. (2016). E-SECO ProVersion: An Approach for Scientific Workflows Maintenance and Evolution. Procedia Computer Science, 100, 547-556.

[3] Missier, P., Belhajjame, K., & Cheney, J. (2013). The W3C PROV family of specifications for modelling provenance metadata. In Proceedings of the 16th International Conference on Extending Database Technology. p. 773-776. ACM.

[4] Davidson, S. B., & Freire, J. (2008). Provenance and scientific workflows: challenges and opportunities. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data. p. 1345-1350. ACM.

[5] Moreau, L., Freire, J., Futrelle, J., McGrath, R. E., Myers, J., & Paulson, P. (2008). The open provenance model: An overview. In International Provenance and Annotation Workshop. p. 323-326. Springer Berlin Heidelberg.

[6] Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., & Zhao, J. (2013). Prov-o: The prov ontology. W3C Recommendation, 30.

[7] Marinho, A., Werner, C. M. L., & Murta, L. G. P. (2009). ProvManager: uma abordagem para gerenciamento de proveniência de workflows científicos. In Workshop de Teses e Dissertações em Engenharia de Software, XXIII SBES (Vol. 14). (in portuguese)

[8] Tan, W. C. (2004). Research Problems in Data Provenance. IEEE Data Eng. Bull., v. 27(4), p. 45-52.

[9] Veregin, H., & Lanter, D. P. (1995). Data-quality enhancement techniques in layer-based geographic information systems. Computers, Environment and Urban Systems, v. 19(1), p. 23-36.

[10] Dalpra, H. L., Costa, G. C., Sirqueira, T. F., Braga, R., Werner, C. M., Campos, F., & David, J. M. N. (2015). Using Ontology and Data Provenance to Improve Software Processes. In Proceedings of the Brazilian Seminar on Ontologies (pp. 10-21).

[11] Nascimento, N. M., Viana, M. L., Lucena, C. J. P. (2016) An IoT-based Tool for Human Gas Monitoring, XV Congresso Brasileiro de Informática em Saúde, p. 96-98.

[12] Coelho, R., Cirilo, E., Kulesza, U., von Staa, A., Rashid, A., & Lucena, C. (2007). Jat: A test automation framework for multi-agent systems. In IEEE International Conference on Software Maintenance. p. 425-434.

[13] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, v.29(7), p. 1645-1660.

[14] Doukas, C. (2012). Building Internet of Things with the ARDUINO. CreateSpace Independent Publishing Platform.

Systems Implantation: A Comparative Study and Identification of Gaps in Usual Methodologies

Marisa Panizzi[1,2,3], Alejandro Hossian[4], Ramón García-Martínez[†3,5]
[1] PhD Program on Computer Science, National University of La Plata, Argentina
[2] Information Systems Engineering Department, Technological National University at Buenos Aires, Argentina
[3] Information Systems Research Group, National University of Lanus, Argentina
[4] Science Department, Technological National University at Neuquen, Argentina
[5] Scientific Researchs Comission - CIC Bs As, Argentina
marisapanizzi@outlook.com, alejandrohossian@yahoo.com.ar, rgm1960@yahoo.com

***Abstract*— Systems implantation is one the phases of the software process where the human interaction dimension becomes relevant. However, weakness in the supervision of this dimension is a factor that in many cases leads to the failure of the Project. In this work, systematization and control of the implantation activities in the following methodologies is reported: IEEE 1074, Métrica V3, Rational Unified Process, and SCRUM. An instrument for their comparison is proposed, including the rationale for the analytical dimensions used, and a preliminary interpretation of results is presented emphasizing gaps.**

***Keywords*-Component; Software processes, human aspects, systems implantation, comparative analysis of processes.**

I. INTRODUCTION

The evolution in software engineering has provided an array of methodologies and standards, including IEEE 1074 [1], Métrica version 3 [2], Rational Unified Process [3], Scrum [4][5]. In spite of this evolution, authors like Vasconcelos and Werner [6] argue that there are many problems associated to the description of processes in the existing standards or software process models. In [7], the following issues related to software process models are mentioned: (i) there is no definition of a model that comprises the joint representation of the processes, products, people and organization, (ii) aspects like the organization of work, people and their interactions are not formalized in the software process, and (iii) there is no defined process including both the technical and the human part of the process in the model.

The Standish Group Report [8] presents a series of factors influencing the success or failure of a project among which are users' involvement, inaccurate definition of requirements, and specialized resources, all of them related to the human aspects of software development.

The improvements proposed by Requirements Engineering such as IEEE-830 [9], 29148-2011 standards [10] for requirement specification, and the Language Extended Lexicon [11] have helped to resolve issues related to the human aspects of the first link in the software construction chain.

DOI reference number: 10.18293/SEKE2017-012

We argue that the implantation process constitutes another link in the software process chain where strong human interaction between systems professionals and the user community is observed. This process is played down because it is the last link in the software process chain. Such underestimation of the process is reflected in the fact that the professional who is assigned the role of implementer or responsible for the delivery of the software product to the customer does not possess the socio-technical competences required to work on such process.

This work was developed on the assumption that software projects fail due to human aspects, roles definition, the interaction between the different roles, the capacities of people performing those roles, among other factors. We state the human dimension and its impact in the implantation process as the working hypothesis. From this hypothesis, the following research question arises: Will a comparative study show that all areas in the information systems implantation process are covered?

This article presents the processes considered (Section II.), the dimensions considered for analysis are described (Section III), results from the comparison are presented along with preliminary interpretations (Section IV), and conclusions and future lines of work (Section V) are provided.

II. PROCESSES CONSIDERED

In this section, the methodologies included in the comparative study are presented: the IEEE 1074 Standard [1], the Métrica version 3 methodology [2], the Rational Unified Process [3], and finally the SCRUM methodology [4][5].

III. DIMENSIONS CONSIDERED IN THE ANALYSIS

This section presents the dimensions considered in the analysis of the implantation process of an information system as well as the project management process associated to it. The dimensions considered are the phases, activities and tasks that make up a process, the proposed tools, the proposed techniques, the artifacts (inputs and products), the required

roles and the competencies needed for each role.

IV. Results and Interpretation

This section presents a table comparing the dimensions of analysis (section A) and the interpretation of the results obtained (section B).

A. Comparative Table of the Dimensions Involved

Table 1 presents the level of compliance of the analytical dimensions considered for the implantation process in the software development process (see Implantation in the Table) and in the software project management processes (see Project Management in the Table).

In this work, we focused on the implantation process within the software development process. It was also necessary to consider the software project management aspects due to the interactions observed with the implantation process.

Table 1. Comparative Table of the Analytical Dimensions.

	Methodologies/Processes							
	IEEE 1074		METRICA Version 3		Rational Unified Process		SCRUM	
	Implantation	Project Management	Implantation	Project Management	Implantation	Project Management	Implantation	Project Management
Phases/ Activities/ Tasks	■	■	■	■	■	■		■
Proposed Tools					■	■		■
Proposed Techniques			■	■	■			■
Artifacts (Inputs/ Products)	■	■	■	■	■	■		■
Required Roles			■	■	■	■		■
Competences								

Since terminology for the two processes analyzed and the analytical dimensions considered have been unified in the table above, it is necessary to describe the terminology unification process. In the IEEE 1074 standard [1], the implantation process is referred to as installation process and the project management process is included in the project management projects and integral processes. The standard refers to the input information and the information resulting from the processes. In the table above, we called this information "artifacts" (inputs/products).

In the Métrica version 3 [2] methodology, the equivalent term for the implantation process is "Implantation and User Acceptance Phase". The equivalent term for project management is addressed in the Project Management interfaces. Under this methodology, the concept of roles is referred to with the term "participants".

In the Rational Unified Process [3], the equivalent concept for the implantation process is included in the deployment flow and transition phase while the project management process is addressed in the Project Management Flow.

The Rational Unified Process introduces the concept of worker, which, in our analysis, will be included in the Required Roles dimension.

The SCRUM methodology [4][5] does not include a process equivalent to implantation. In turn, the project management process is addressed under that same name.

B. Interpretation of Results

The results of the analysis of the Comparative Table of the Dimensions Involve (section 4.A.) allowed us to draw the partial conclusions addressed in the following paragraphs.

In relation to the Phases/Activities/Tasks dimension, the IEEE 1074 standard [1], the Métrica version 3 methodology [2] and the Rational Unified Process [3] propose phases, activities and tasks for both process. In the SCRUM methodology [4][5], activities are defined for the project management process but there are no activities for the implantation process.

With regard to the proposed Tools, the IEEE 1074 standard [1] and the Métrica version 3 methodology [2] do not propose any tools for the activities in either process while the Rational Unified Process [3] proposes tools for both processes. The SCRUM methodology [4][5] proposes tools of software project management process.

As for the Proposed Techniques section, the IEEE 1074 standard [1] does not propose any techniques for either process while the Métrica version 3 methodology [2] proposes techniques for both processes. In turn, the Rational Unified Process [3] proposes techniques for the implantation process, and the SCRUM methodology [4][5] proposes techniques for the software project management process.

In relation to the Artifacts (Inputs/Products) dimension, the IEEE 1074 standard [1], the Métrica version 3 methodology [2] and the Rational Unified Process [3] propose artifacts for both processes while the SCRUM methodology [4][5] proposes artifacts for the software project management process only.

With regard to the Required Roles dimension, the IEEE 1074 standard [1] does not propose roles for any of the studied processes. The Métrica version 3 methodology [2] and the Rational Unified Process [3] present roles for both processes while the SCRUM methodology [4][5] proposes roles for the software project management process only.

As for the Competences dimension, none of the methodologies considered include competences for the processes.

V. Conclusions and Future Lines of Work

This work presented a systematic revision of the system implantation sub-process by means of a comparative study of the selected methodologies. As a result, a number of gaps in the sub-process were identified, particularly gaps related to human aspects. Even though there are models that propose roles, the competences that the people playing those roles should have are not presented.

We were able to create a comparison instrument to study a number of analytical dimensions of the implantation process and the project management process that interacts with it.

Future lines of work include: (a) the need to incorporate more process models or methodologies, such as: Extreme Programing (XP), Dynamic Systems Development Method (DSDM), Rapid Application Development (RAD), Agile Unified Process (AUP); and b) to explore the incorporation of other dimensions to the comparison tool, such as metrics.

ACKNOWLEDGMENT

The research presented in this paper has been partially funded by: Research Project UTNBA4347 of Technological National University at Buenos Aires, Research Project 80020160400001LA of National University of Lanús and Research Project PIO CONICET-UNLa 22420160100032CO of National Research Council of Science and Technology (CONICET), Argentina. The translation to the English version of the paper was funded by the Information Systems Engineering Department of the Technological National University at Buenos Aires, Argentina.

REFERENCES

[1] IEEE 1074, 1997. IEEE Standard for Developing Software Life Cycle Processes. IEEE Std 1074 (Revision of IEEE Std 1074-1995; Replaces IEEE Std 1074.1-1995) (1997).

[2] PAe, Métrica versión.3. Portal de Administración Electrónica. Gobierno de España.

[3] IBM. Rational Software. Péraire C., Edwards M, Fernandes A., Mancin E. y Carroll K. The IBM Rational Unified Process for Systems (2007).

[4] Scrum Manager. Palacio J. Gestión de Proyecos SCRUM Manager (Scrum Manager I y II). http://www.scrummanager.net. Página vigente al 17/06/2016 (2015).

[5] Schwaber K. y Sutherland J. La Guía Definitiva de Scrum: Las reglas del juego. (2013)

[6] F. M. de Vasconcelos Jr. and C. M. L. Werner, "Software development process reuse based on patterns". Proceedings of the Ninth International Conference onbSoftware Engineering and Knowledge Engineering (June 1997) pp. 97- 104 (1997).

[7] Acuña S., Juristo. N. Moreno A., Mon A. A software Process Model Handbook for incorporating people `s capabilities. United States of America: Springer Science+Business Media, Inc. (2005).

[8] Report, The Chaos Manifesto: Think Big, Act Small. Stadish Group. (2013).

[9] IEEE 830. IEEE Recommended Practice for Software Requirements Specifications - IEEE Std 830-1998. (1998).

[10] ISO/IEC/IEEE 29148. Software & Systems Engineering Standards Committee of the IEEE Computer Society, ISO/IEC/IEEE 29148:2011(E), Systems and software engineering — Life cycle processes — Requirements engineering (2011).

[11] LEITE JCSP. Engenharia de Requisitos, Notas Tutoriales, Teaching material of the Requirements Engineering course, Computer Science Department of PUC-Rio, Brazil (1994).

Custom Process to Small Business

Rodrigo Rocha Silva
FATEC-SP
CISUC
Mogi das Cruzes, Brasil
rodrigo.rsilva@fatec.sp.gov.br

Fernanda Yuri Kimura
FATEC-SP
Mogi das Cruzes, Brasil
feerkimura@hotmail.com

Jorge Bernardino
Polytechnic of Coimbra
CISUC
Coimbra, Portugal
jorge@isec.pt

Joubert de Castro Lima
UFOP
Ouro Preto, Brasil
joubert@iceb.ufop.br

***Abstract*— In this work we customized the RUP process with Scrum practices, and proposed a differentiate traceability matrix, applying in a small company. The experimental results show that our customization can be adopted as an alternative to a systematic and less-intrusive process.**

Keywords*— *Software Development Process, Scrum, RUP

I. INTRODUCTION

Some practitioners and academics have proposed different process that combines two or more methods [1, 2]. The objective of these approaches was to create a process that maximized the strengths of the involved methods while at the same time reducing their weaknesses to improve the software development lifecycle and produce high-quality products [1].

The main goal of our custom process is to guarantee a constant delivery routine of software artifacts that have business value. In order to validate our custom process, we present the results of its adoption in a small software company.

II. OUR CUSTOM PROCESS

Our custom process fulfills all the basic premises of software development processes, such as iteractions and verifications with delivery forecast of mandatory artifacts.

The customization proposed is strongly focused on design and development, suggesting only the following RUP artifacts: I. DVS or technical proposal for high-level definition of the scope and purpose of the system; II. Mapping of functional requirements; non-functional; and business rules; for specification of functionalities, presenting the constraints, validations, and exceptions that the system must obey; III. Description of functionalities; IV. Software Domain Mapping.

Our custom process defines three roles similar to the *Scrum* roles, for which each description can be found below: I. Product owner; II. Guardian; III. Team. Our custom process is divided into three phases, just like in the Scrum, although it receives the name of the RUP phases: *Initiation, Elaboration* and *Construction*, that compose the development *Sprints* and the *Transition* phase.

In order to ensure the control of requirement changes, our custom process suggests the creation of a traceability matrix. The implementation of test units with the greatest possible coverage is a key practice in using our custom process.

III. RESULTS

Our custom process was applied in a real project of Company X, where typically, the projects are developed by a team of 3 to 5 people. Prior to our custom process application, the software development process was based on informality, without documentation of requirements and scheduling of the next meetings. Most of the times, deadlines were not met and there were many changes in the requirements during the development phase because those were not previously foreseen due to the lack of planning during the specification and analysis processes.

As proposed by our custom process, the control of the project was carried out by implementing the culture of sending weekly reports, thus maintaining a communication between the team and the project manager.

In order to validate whether the requirements were developed according to the specification, the developer himself performed the tests on the functionalities developed, which were later validated by the systems analyst. The inconsistencies found were documented by the systems analyst and sent to the developer for correction.

In relation to the schedule, it was estimated four months to end the project. Tasks were completed as planned only in the first two *sprints*. The last three weeks were delayed by two weeks each, which represents 30% of the total planned. The low number (5%) of changes in requirements was also evident.

The adoption of the reporting culture was also one of the new habits that faced difficult adaptation. Resistance to new practices was the greatest difficulty at implementing our custom process.

VI. CONCLUSIONS

The implementation of our custom process was carried out with a compatible cost to small companies. At first, the change of culture generated dissatisfaction in the development team, and it was of great importance their awareness that a defined process is fundamental for the improvement of the final product and also makes the work of all the involved ones, easier.

Future works intend to apply our custom process in other companies as well as in different projects, besides inserting alternatives to the practices for projects with differentiated scopes.

REFERENCES

[1] Nortier, B., K. Von Leipzig, and C. Schutte, “The Development of a Software Development Framework by Combining Traditional & Agile Methods to Address Modern Challenges,” ISEM 2011 Proceedings, September 21-23, Stellenbosch, South Africa, 2011.

[2] Bashir, M. S. and M. R. J. Qureshi, “Hybrid Software Development Approach for Small to Medium Scale Projects: RUP, XP & SCRUM,” Science International (Lahore) 24, 4 (2012), pp. 381-384

DOI Reference Number: 10.18293/SEKE2017-31

Visual Development Platform for Ruby on Rails

Anmol Desai*, Nicholas Molloy*, Jing Sun† and Gillian Dobbie†

*Department of Electrical and Computer Engineering
†Department of Computer Science, The University of Auckland, New Zealand
Emails: *{ades597, nmol886}@aucklanduni.ac.nz, †{jing, gill}@cs.auckland.ac.nz

***Abstract*—This paper briefly reviews the construction of Ruby on Rails applications, identifies the pitfalls of existing tools, and proposes the design and development of a lean cross-platform desktop application, along with an evaluation of the prototype.**
***Index Terms*—Ruby on Rails, Visual tool, Web Applications**

I. Introduction

Ruby on Rails is a framework that runs on the Ruby language, and provides developers the tools necessary to build modern web services [1]. While rails simplifies the creation of web-services, it can be found to be difficult for a beginner to not only install the framework, but to also learn and use it. This in part is due to the fact that the developer interacts with rails predominantly through the command line. This paper presents the initial research on existing tools, the design decisions, the development and implementation of a visual development platform for Ruby on Rails, namely the *Rails Editor*.

The tool provides the developers with many features that aim at enhancing the productivity of a web application in Ruby on Rails [2]. The developers are able to execute many Ruby on Rails commands visually, clone from a git repository, create model/controller/routes diagram, visualise database schemas, chat and use a native inbuilt terminal. The collection of all these features in the editor makes the development of Ruby on Rails projects much more convenient as well as efficient.

II. Design and Implementation

The *Rails Editor* was developed as a cross-platform desktop application, which utilises and extends the Electron framework, Chromium, Node.js, HTML, CSS, etc. Figure 1 below shows the overall architecture of our solution.

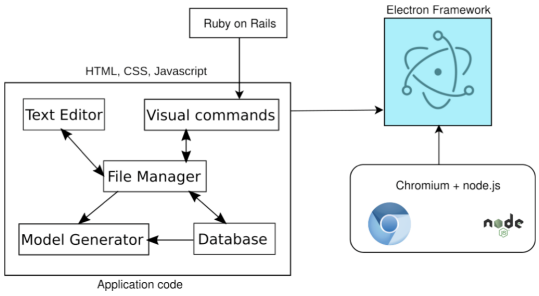


Fig. 1. Overall architecture of the tool

DOI reference number: 10.18293/SEKE2017-047

In order to aid in developer's understanding of Rails projects that they are working on, several diagrams, e.g., the model, controller and route diagrams, can be generated and viewed in the tool. These diagrams illustrate the important aspects of a Rails project in a visual manner. For example, Figure 2 shows an example of the generated route diagram.

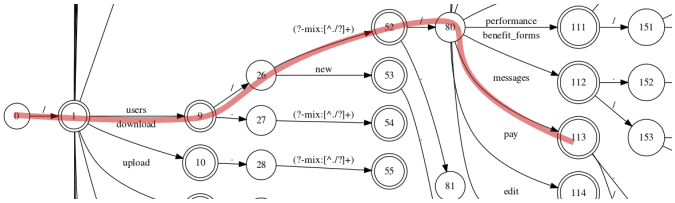


Fig. 2. An example of a route diagram generated by the tool

III. Evaluation and Conclusion

Three types of evaluations were conducted. A user survey showed that all of the features implemented were received positively by the participants. A comparison with existing tools showed that the tool developed implements many features that are not existent in currently available tools. A small performance test showed that the tool is able to perform within reasonable limits. Finally, it is worth mentioning that the *Rails Editor* project won the Final Year Research Project Prize in the Software Tools category by a panel of judges from ICT industries in 2016.

In the future, we plan to make the following improvements. The chat feature could be extended to support a group of developers. Currently chat only allows two users to communicate at a time. It could be extended to allow team programming features, in a similar way to products such as Skype or Google Hangouts. Since the tool is implemented using web technologies, a large portion of it can be run within a standard web browser. The application could be modified and extended in such a way that it runs on a "cloud" infrastructure. Features such as running rails commands and the terminal would be executed remotely, allowing a user to access their development environment from any where using a web browser.

References

[1] B. Tate and C. Hibbs, *Ruby on Rails: Up and Running, 2nd Edition*. O'Reilly Media, October 2008.
[2] C. Jones, *Applied Software Measurement (2Nd Ed.): Assuring Productivity and Quality*. Hightstown, NJ, USA: McGraw-Hill, Inc., 1997.

Home Automation: HMM based fuzzy rule engine for ambient intelligent smart space

Gopal Singh Jamnal
School of Computing
Edinburgh Napier University
Edinburgh, UK

Xiaodong Liu
School of Computing
Edinburgh Napier University
Edinburgh, UK

Lu Fan
Payfont Ltd.
Edinburgh, UK

***Abstract*—in this paper, we proposed a new type of decision-making system to achieve the intelligent goal for automated smart environments. The artificial intelligence techniques, used as building blocks to understand inhabitant activity patterns. The collected information fused to a central inference engine based on Hidden Markov model and Fuzzy rules for taking appropriate actions to communicate and control various home appliances. We proposed a novel CASH (cognitive automated smart home) architecture, based on the Hidden Markov Model and the fuzzy rule based system. The Hidden Markov Model and fuzzy rules are well equipped to address the spatio-temporal activity pattern recognition problem and to trigger appropriate task execution rules.**

Keywords— home automation; rule mining; smart home; artificial intelligence; expert system; cyber physical system.

I. INTRODUCTION

The rapid use of interconnect network objects such as embedded-sensor, RFIDs, BSN are in trend. These interconnected technology known as cognitive Internet of Things (CIoT). Artificial intelligence provides foundation as build blocks to design automated IoT systems. The aim of an ambient intelligent system is to provide comfortable assisted living to inhabitants. Scientific work has been done in ambient intelligent space such as Care-lab, CASAS, Grator-Tech HIS, aware home, iDorm and MavHome projects, the process of activity recognition is subdivided into four part as (i)sensing, (ii)data-preprocessing, (iii)data modelling for feature extraction (iv)feature selection [1]. Many blue-chip companies including IBM Watson, researching into identifying inhabitants preference, activity patterns to provide a customized digital assistant for granting access and controlling various appliance to automated tasks[2].

For recognizing inhabitant activity patterns in smart home scenario, Hidden Markov Model plays a key role. HMM is a generative probabilistic model used for identifying hidden states <*s1,s2,..,sn*>from given observation sequences<*o1,o2,..on*>[3]. Furthermore, [1] suggested that observable state sequence*(st1)* at time *t* , depends only on the current state*(st)*, irrespective of previous state *(st-1)*.During HMM training we tend to find optimal state sequence with higher probability of *Pr(S|O)*, known as most likelihood pattern. As an iterative trend, HMM require re-estimation of input parameters to train the system (transition and emission matrix), Baum-welch algorithm work well here as solution. As a result, such re-adjusted parameters increase the probability of finding optimal state sequence. At a later stage, Viterbi algorithm works to find most likelihood patterns.

Furthermore, in later stage, adaptive fuzzy rules have been applied for task executions. The rule based system represents the human expert's knowledge, encoded in set of rules that tell system what to do and what to accomplish in different activity situation as set of if-then statements*(if activities(a1, a2 and a3 is active) – then rules(r1,r2 and r3 activated).*[4].

II. CASH (COGNITIVE AUTOMATED SMART HOME) FRAMEWORK

The proposed CASH architecture, works in two components: (i) recognizing inhabitant activity from sensor's data, (ii) task execution through fuzzy rules activation. In first component, embedded sensors capture activity sequences of inhabitants and identify Hidden likelihood patterns of activities. While in second component, fuzzy rule adaptation applied for task execution, the set of likelihood activity patterns mapped with fuzzy rules. Fuzzy rules invoke operations on home appliances for switching On/Off, set temperature etc.

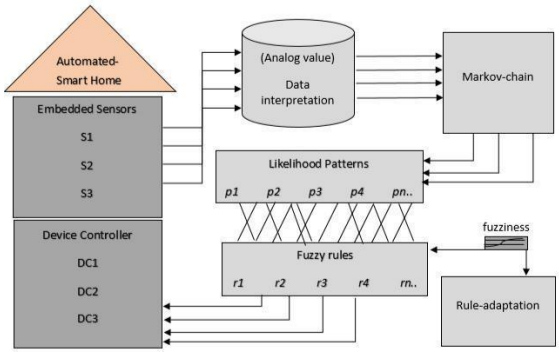


Figure 1. The proposed CASH (cognitive automated smart home).

[1] DOI reference number: 10.18293/SEKE2017-035

In the use case scenario, we placed 9 sensors capturing light, motion, noise inside a studio flat. These sensors are well programmed with Arduino Uno Wi-Fi micro-controller and placed appropriately inside bedroom, living room and kitchen area. Hidden Markov model analyze datasets to identify daily routine activity sequences of inhabitant. The proposed CASH model, trained over captured data sets for multiple time using Balm-welch and Viterbi algorithms to obtain most likelihood pattern of activity sequence. As human activity are overlapped and fuzzy in nature, therefore applying fuzzy rules is right solution to the problem. Fuzzy rules are applied on identified activity pattern to control various appliances inside home for providing a comfortable automated living to inhabitant.

Table 1. Sensor's digital state for activity labeling

time-sequence	*t*	*t+1*	*t+2*	*t+3*	*t+4*	*t+5*	*t+6*
activity							
reading	0	0	1	0	0	1	0
Woking	1	1	1	1	1	1	1
cooking	0	1	0	0	0	0	1
Sleeping	0	0	0	0	0	0	1

The nine sensors, capturing motion *(m)*, light *(l)* and noise *(n)*, have given unique ID for data interpretation. The motion sensors, track inhabitant mobility so placed in center and light sensor placed in the corner area while sound sensor placed near by TV set. All sensors are connected to Arduino UNO Wi-Fi microcontroller as access point.

In CASH framework, fuzzy rules linked up with activity patterns as <*antecedent*> and <*consequent*> manner. In table 2, various rules defined to automate appliances in smart home for providing a comfort living for inhabitants. Those fuzzy rules provide <*consequent*> interface for inference engine such as *IF* (likelihood pattern *p1, p2, p3* identified) *THEN* trigger rules (*R1, R2, R3).*

Table 2. Automated tasks as fuzzy rules.

ID	Fuzzy rule	Automated task for activity
1	R1	Tea kettle on→lock doors→set temp. →switch off TV→ switch on study lamp
2	R2	Switch on TV favorite channel, set temp. →dim light
3	R3	Warm up oven →tea kettle ON → ask for laundry/dishwashing
4	R4	Lock doors→set temp→control blinds → switch off TV
5	R5	Set temp→ tea kettle on→study lamp ON

We used 250 data sets of observed sequence over three days from 01/01/2017 to 03/01/2017, which are labelled based on each activity type. The training set and testing set have been divided into 180 and 70 sets respectively. Matlab 2016 version were used for experiments, furthermore 180 observed activity datasets have been labelled into four main activities. Initial transition and emission matrix have been defined based on prior knowledge base. Afterwards, training and testing performed using Matlab's *hmmestimate* and *hmmtrain* methods, Baum-welch algorithm applied for further re-estimation of transition and emission probability matrix parameters and train system with new parameters. Matlab experiments trained the system over 180 observed sequences. Finally, *hmmviterbi* methods used to apply Viterbi algorithm to identify most likelihood activity sequences. Using 80 testing activity datasets, system achieve 79% of accuracy, shown in fig. 2. as Matlab simulation result.

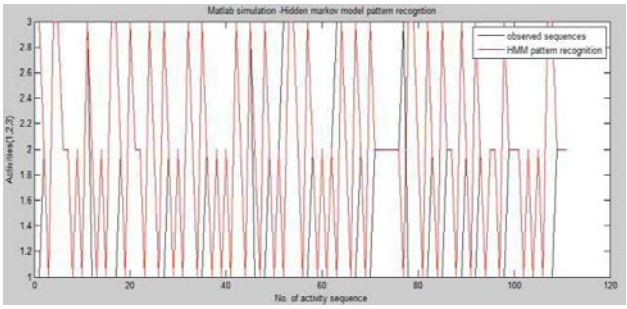


Fig. 2. HMM likelihood activity sequence compares to observedsequence.

III. CONCLUSION

This paper proposed a new type of CASH (Cognitive Automated Smart Home) architecture to provide comfortable inhabitant daily living by understanding their needs and activity intention. We embraced the knowledge base from traditional context aware pervasive computing and combined with the Cognitive Internet of things. Furthermore, the building block of CASH architecture is well equipped with artificial intelligence machine learning algorithms of Hidden Markov Model and fuzzy rules-based system. CASH recognize hidden pattern in daily activity living and apply appropriate rule for task execution. Moreover, CASH framework, can be applied in elderly health care system, smart classrooms, and smart spaces for automated environment

REFERENCES

[1] G. Jamnal and X. Liu, "A cognitive-IoE approach to ambient-intelligent smart home," IoTBDS 2017., in press.

[2] Ibm.com, 'IBM Watson Internet of Things(IoT)', [2017]. [Online]. Available: https://www.ibm.com/internet-of-things/. [Accessed: 01-Apr-2017].

[3] M. R. Hassan, K. Ramamohanarao, J. Kamruzzaman, M. Rahman, and M. Maruf Hossain, "A HMM-based adaptive fuzzy inference system for stock market forecasting," *Neurocomputing*, vol. 104, no. July 2016, pp. 10–25, 2013.

[4] C. Grosan and A. Abraham, "Rule-Based Expert Systems," *Intell. Syst.*, vol. 17, pp. 149–185.

Named Entity Extraction and Classification in Digital Publications

Chuan-Yu Wu*, Bom Yi Lee*, Jing Sun†, Yin Yin Latt‡, Kim Shepherd‡ and Jared Watts‡

*Department of Electrical and Computer Engineering
†Department of Computer Science
‡Library Digital Services
The University of Auckland, New Zealand
Emails: *{cwu323, blee660}@aucklanduni.ac.nz, †j.sun@cs.auckland.ac.nz, ‡{y.latt, k.shepherd, j.watts}@auckland.ac.nz

***Abstract*—This paper describes the design and implementation of a PDF extraction tool, which provides the functionalities of meta-data creation, bibliography extraction, HTML conversion and key phrase classification. Evaluation results showed high accuracy rates and good performance measurements.**
***Index Terms*—Text Extraction, Automatic Classification**

I. Introduction

The University of Auckland is New Zealand's largest research organisation, with over 13,000 staff and postgraduate students involved in fundamental and applied research. Its General Library holds thousands of publications and dissertations within its repository. The standard file format for publications is in Portable Document Format (PDF), which is not content-accessible [1]. This means that, although the library repository holds a large amount of information, this available information is not being utilised to its full potential.

II. Design and Implementation

To enable automated extraction of information from the digital scholarly publications in the library repository, it involves analysing and extending existing tools that fit our purpose. Our project has explored four areas of information extraction and classification from publications in PDF format. We have developed separate tools for each feature, which will be used by the library. The general structure of our overall solution is shown in Figure 1.

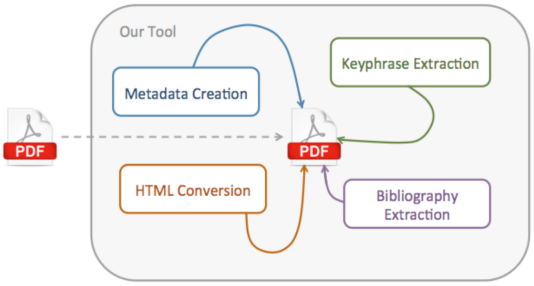


Fig. 1. Overall Structure of the PDF extraction tool

DOI reference number: 10.18293/SEKE2017-132

Four main components included in our project are metadata creation, key phrase extraction, HTML conversion and bibliography extraction. These features were implemented by utilising and extending the existing tools, such as the Apache PDFBox, KEA, PDF2Dom and Freecite. Our keyword identification uses the ACM Computing Classification System SKOS file. The front-end GUI is a means of displaying the resulting output of each tool. It was implemented using JavaFX with SceneBuilder following a model-view-controller (MVC) architecture.

III. Evaluation and Conclusion

The developed tool was evaluated against quantitative and qualitative analyses to measure its efficiency and viability for use within the library repository. We have carried out our evaluation on a set of 30 open-source publications that were provided by the library team. The results showed high accuracy rates, and fast processing time in data extraction and key phrase classification. A comparison table was drawn up to compare the features provided by several existing tools in relation to our solution, as shown in Figure 2.

	Open source	PDF conversion	Keyword Extraction	HTML conversion	Bibliography Extraction	Standalone	Cross-platform
pdftotext	✓	✓				✓	
PDFlib (TET)		✓				✓	✓
PDFMiner	✓	✓				✓	✓
KEA	✓		✓			✓	✓
PDFBox	✓	✓		✓		✓	✓
FreeCite	✓				✓	✓	✓
Our tool	✓	✓	✓	✓	✓	✓	✓

Fig. 2. Feature comparison between our tool and exisiting tools

Our project has taken a breadth over depth approach to explore the different solutions. Further extensions can be made by incorporating: (1) Machine learning process during key phrase extraction; (2) Deriving sets of characteristics for disciplines other than Computer Science to allow extraction of entities from any types of scholarly publications.

References

[1] D. D. A. Bui, G. Del Fiol, and S. Jonnalagadda, "Pdf text classification to leverage information extraction from publication reports," *J. of Biomedical Informatics*, vol. 61, no. C, pp. 141–148, Jun. 2016.

Knowledge Management in a Software Development Organization: Identifying Tools, Processes and Benefits

Felipe Furtado
CESAR
Recife, PE, Brazil
fsfs@cesar.org.br

Gustavo Alexandre
CESAR
Recife, PE Brazil
ghsa@cesar.org.br

Nelson G. de Sá Leitão Júnior
Informatics Center
UFPE University and
CESAR
Recife, PE, Brazil
leitaojr@outlook.com

Ivaldir de Farias Junior
Informatics Center
UFPE University and
São Miguel College
Recife, PE, Brazil
ivaldirjr@gmail.com

Hermano P. de Moura
Informatics Center
UFPE University
Recife, PE, Brazil
hermano@cin.ufpe.br

Organizations usually seek for competitive differentials by investing money, time and effort usually in aspects such as information and communication technologies, lowering operating costs, improving productivity and the quality of their products. However, in late years, the importance that is given to the internal organizational knowledge has been increasing, either to understand the characteristics and demands of the competitive environment as well as to understand the needs associated with the organizations' processes.

This context demands an internal Knowledge Management approach in the organizations. Knowledge management (KM) can be defined as the process of capturing and utilizing the collective expertise of an organization in whatever the activities of its business.

This work has the objective of identifying the KM tools and processes used by a Brazilian private institution that creates products, processes, services and innovative companies using Information and Communication Technologies, as well as to identify the perceived benefits in the form of examples of success KM scenarios in different organizational contexts.

The data collection was carried out in November and December 2016 and had adhesion of 33 employees distributed in different areas: administrative, human resources, entrepreneurship, marketing, financial, operations, internal IT support, among others.

Existing initiatives at the institution were classified into processes and tools, and each of the participant answered the following question: "What tools / processes are used or have been used by your area for knowledge management?" The following results were found: Google Drive (94%), e-mail lists (81%), Slack (65%), lessons learned meetings (65%), Scrum retrospective meeting (50%), Basecamp (38%), PMO website (38%), SVN (38%), Dropbox (38%), internal sites (31%), Yammer (25%), Wiki (25%), CVS (19%), internal social network (19%), Slideshare (19%), Youtube channel (16%), Moodle (16%), CRM (9%), and others (41%).

Among the most used tools, we can highlight the use of Google drive, email lists and Slack. Additionally, among the "Other" options, the following tools / processes were listed by the participants: internal meeting for organizational news, technical seminaries, lessons learned meetings through Hangouts, Zoom, Skype and Confluence, internal mailing for organizational news, Mind Maps, Keepass, BI and Trello.

Although Google Drive has been pointed out by 94% of respondents, it has been widely used to capture and store knowledge or simply to perform backups of files. In some cases, it has worked only as a repository of reports, experiences, and techniques. There was no organizationally effective way for the remaining stages of the KM process: to refine, manage and disseminate knowledge. In a somewhat smaller proportion, but in a similar way, the e-mail lists have been used by most respondents in the sense of disseminating knowledge, but in a non-systematic way.

Regarding the main results achieved within the areas with the use of these tools and processes, we can cite some examples: support in the team allocation process to identify collaborators with certain technical knowledge, sharing information and building artifacts collaboratively, repository of documents in a structured way, facilitating the search and future analysis, either for legal necessity or as support in future similar projects, knowledge transfer in technologies and reduction of dependency on a single person who holds knowledge.

From the perception of participants, we conclude that, in general, the tools and processes used by the institution have helped the organization in the improvement of communication and diffusion of knowledge, but without a systematic process of knowledge management. The Institute is always in search for the improvement of the KM. Still, with the identified usage numbers of KM tools and processes, and the specific usage of some approaches by specific organizational departments, we noticed a tendency of dispersion.

Access to information in the organization is still difficult, but several participants indicate that it has already improved with the use of Slack. It is still perceived that there is a sense of silos, and the fact that some Non-Disclosure Agreements contracts with part of the customers, also prevent further sharing of information. Some specific knowledge is still in one or a few people, and when they leave, knowledge is lost. Future works include performing this survey in multiple organizations, to compare and analyze the results.

DOI reference number: 10.18293/SEKE2017-163

Implementing and Evaluating Scrum in Computer Science Senior Projects

Maral Kargarmoakhar
Computer Science Department
Florida International University
11200 SW 8th St
Miami FL 33199
mkarg002@fiu.edu

Mohsen Taheri
Computer Science Department
Florida International University
11200 SW 8th St
Miami FL 33199
mtahe006@fiu.edu

S. Masoud Sadjadi
Computer Science Department
Florida International University
11200 SW 8th St
Miami FL 33199
sadjadi@cs.fiu.edu

***Abstract* - This empirical study examines the adoption of agile software development, and the role of Scrum in computer science senior projects at Florida International University. This paper describes the senior projects and Scrum implementation. It highlights the advantages of incremental and iterative software development and discusses how Scrum can improve the productivity of software teams. In addition, it illustrates the other benefits such as engagement, transparency, frequent delivery and flexibility to change. Finally, it evaluates the outcomes by tracking the velocity estimations of each Scrum team. It demonstrates how Scrum and our tools can facilitate the software product development and the transition of our projects throughout semesters.**

I. INTRODUCTION

Agile software development (Agile) has been a trending topic in the software engineering during the past decade [1, 2]. Learning and implementing agile software development at school is challenging. Students not only need to work as a team but also must commit to the principles of Agile. The proposed solution in this paper describes the process of selecting projects, assignments, and software development management. Our solution is useful for software instructors at high school and university levels especially for those who assign semester projects to students.

In this paper, we show how we adopted Scrum on a large scale for several teams during two semesters at Florida International University (FIU) that complements the previous research. Our computer science (CS) students at the beginning of the semester join their favorite projects. They learn and practice Scrum in a systematic way, and they work with some best practice Scrum tools that we either adopted or developed ourselves. Benefits of adopting Scrum methodology for bachelor CS senior students are evaluated in this study.

II. SCRUM

Agile software development is an umbrella term for several iterative and incremental software development methods that follow agile manifesto [3]. Agile manifesto values individuals and interactions over processes and tools. Agile concentrates on working software, customer collaboration and rapid response to change [4, 5].

Scrum is a lightweight framework that emphasizes teamwork. It provides an iterative progress toward a well-defined goal [6]. A scrum team includes developers, a scrum master, and a product owner. A product owner provides a prioritized wish list of user stories called a product backlog. The team has a certain amount of time called sprint, which is usually two to four weeks. During a sprint planning meeting, the team selects several user stories from the backlog and then decides how to implement those user stories. Each team has a Scrum master who keeps the team focused on its goal and facilitates the meetings.

III. EXPERIMENT

Our senior projects are mostly focused on the software application development. With our wide advertisement at school and in Miami, FIU faculty members and their industrial partners become aware of this opportunity and submit their projects in our website (vip.fiu.edu). They need to propose their project at least one month before the semester. We evaluate the projects carefully and approve the qualified ones.

At the beginning of the semester, we introduce all projects to students. We provide them with a link to the last semester's deliverables including the main Github repository of the projects (https://github.com/FIU-SCIS-Senior-Projects). This repository consists of introductory videos, PowerPoint presentations, and all the code and documentation. We use one of our own tools (http://spws.cis.fiu.edu) to match students with the projects automatically based on students' skills and the skills required for each project. The course instructor is in touch with students and product owners to make sure students get assigned

DOI reference number: 10.18293/SEKE2017-183

to the right projects. Then we form the teams with three to seven members

To make a bridge between semesters, and a smooth transition, students need to make several videos for the next semester students. These videos not only are the introduction to the project, but also give an overview all of the complete user stories, a wish list of future features and an installation guideline. Students upload their videos to YouTube senior project channel and make a playlist of them in the FIU CIS senior project website (http://seniorproject.cis.fiu.edu).

In addition, we use many other useful tools such as Git and Github for version control, Skype and GoToMeeting for online meetings, and Slack for on-demand team communication. In our study, during Spring and Summer 2016, we tracked students' progress. We taught them how to assign story points to each user story based on working hours and the complexity of each user story. In the next section, we introduce story points, velocity and the evaluation process.

I. EVALUATION

For our experiment, we asked students to reflect their sprint planning in Google drive and Mingle [7] (https://fiu-scis-seniorproject.mingle.thoughtworks.com). Thus, we had access to all the information about implemented user stories. In the Spring semester, we had 15 teams and 45 students. The number of students in each team varied from two to six depending on the complexity of the project. During the first sprint we worked with each team one by one and made sure that they used Scrum correctly. We provided the students with several guidelines, and tutorials which allowed them to practice the story points and velocity estimation.

In each semester, students work five to seven sprints (Summer semesters are shorter). During the first week, they try to bring their projects up to speed. The last week of the semester is reserved for preparation of the showcase. All students need to work at least 20 to 25 hours a week depending on their semester. Students use a Google sheet called timecard to reflect their working hours every day. Scheduling the best time for meetings is not always easy. We provide students with a weekly schedule template, a guideline and a Google calendar link to sync their availability with the team.

In our experiment, story points and velocity are two of the main metrics for evaluation. A story point is used to measure the effort required to implement a user story. In other words, it is a number that identifies how difficult the user story is. The raw numbers we estimate are not important, rather the relative values matter. For instance, if we assign four points to user story A, and we assign eight points to user story B, this means that user story B is two times more difficult than user story A.

There are two ways to compare the user stories with the baseline user story. The first way is using the total required hours for completing a user story. The second one is using the level of complexity. In our experiment, we used the second one that means that story points have no relevance to actual hours. It makes it easy for our teams to estimate the points. However, teams identify their hours for each task of a user story in the Mingle as a reference of their actual daily work on each user story. The most common series of points are the Fibonacci series (1, 2, 3, 5, 8, 13, 21, etc.).

We use Scrum planning poker for story point estimation. We believe this game improves the visibility of each user story for all students. To start estimating the story points for each user story, each team picks a well-known user story as a baseline. It is important that the product owner defines all the tasks and acceptance criteria carefully. The team does not necessarily need previous experience on implementing that user story. However, it is essential that team members understand it. The team assigns a random point to this story. Then other user stories have to be sized based on it. A story point estimation must include all tasks involved in getting a user story completed.

Velocity is the total number of story points for all fully "Done" user stories during a single sprint. Velocity is calculated at the end of each sprint. It is important not to compare the velocity between teams. As we described, story point estimation depends on the baseline user story. Teams are likely to have different baseline user stories. Even if they have the same baseline user story, they may estimate different story points for that user story. For example, a team with a velocity of 100 is not better than a team with a velocity of 70.

We tracked all of the velocities for 15 senior project teams in the Spring semester, and 6 teams in the Summer semester. Figure 1 shows the user story points and the growth for the projects. The x-axis shows the number of sprints, and the y-axis shows the number of the total user story points. As some of the students were not familiar with Scrum at the beginning of the semester, we did not consider the first and second sprints in our charts. In addition, the final sprint of the semester was not considered in our charts since students wrapped up their projects and worked on the final deliverables. In Figure 1, the chart start from

number 1 instead of 3 just for better visualization of the process. In this chart, the growth of some of the projects is highlighted with a percentage representation. As the results show, all of the teams made a remarkable progress in implementing the user stories. For example, for the VIP website project, the total story points at the beginning of semester is 25. After four sprints, the total story points are 50 which means they has increased their productivity 100%. Figure 2 shows the total velocity of the teams. Teams have different velocities; but as we described before it does not mean that a team with the highest velocity is better than the other teams.

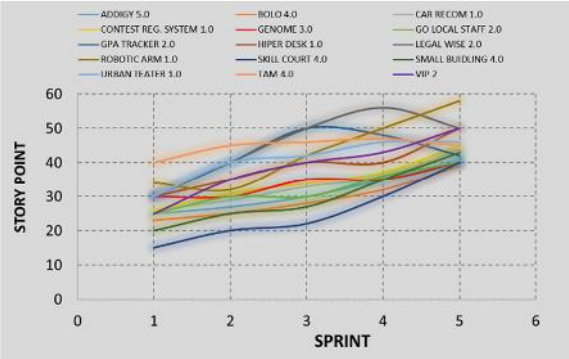


Figure 1. Teams' story points in Spring 2016

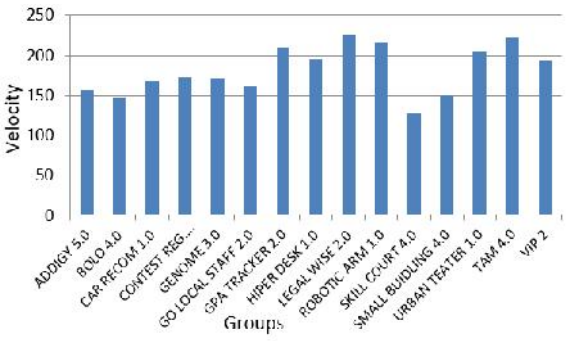


Figure 2. Teams' velocity in Spring 2016

II. DISCUSSION

Transition of the projects between semesters was a challenging task for us. Most of our projects are long-term and have become completed incrementally throughout years. After students submit their final deliverables, we have to transfer the materials to new students. Projects should have a clear and understandable programming structure. This allows the future groups to move forward and develop a new version of the project. If new students spend a lot of time to understand the code, their progress would not be as we initially expected. We use four main tools including Github, Mingle, Google Drive and Youtube to transfer the ownership of a project completely.

For our version of scrum adoption at school, we added another bi-weekly meeting to the semester schedule which is called "Feedback meeting". In this meeting, the instructor plays another role as a mentor. After each sprint, in an in person or remote meeting, the instructor and the team review the user stories together. The feedback meeting helps to keep students focused during the project and resolve some of the inconsistencies. We are still doing Scrum for our senior projects and we will report our new findings soon.

III. CONCLUSION

This study presents the Scrum adoption and evaluation for CS senior projects at FIU. We described the Agile, and Scrum in senior projects. We showed that Scrum helps students significantly in the progress of their project. The short iterations allow teams to deal with uncertainties. We evaluated the teams and showed the improvements. Finally, we discussed the transition of the projects between semesters.

ACKNOWLEGMENT

This research was partly supported by the Leona M. and Harry B. Helmsley Charitable Trust via the Georgia Tech's Vertically Integrated Projects (VIP) Program. The authors also thank Dr. Francisco Ortega, Ravi Kiran Agarthi and Shefali Nakum and the FIU graduate students attended Dr. Sadjadi's 2014- 2016 senior project, VIP, Advanced Software Engineering class for their technical assistance and insightful discussions during the class sessions. This material is based in part upon work supported by the National Science Foundation under Grant Numbers of I/UCRC IIP-1338922, AIR IIP-1237818, SBIR IIP-1330943, III-Large IIS-1213026, MRI CNS-1429345, MRI CNS-0821345, MRI CNS-1126619, CREST HRD-0833093, I/UCRC IIP-0829576, and MRI CNS-0959985.

REFRENCES

[1] Dingsøyr, Torgeir, et al. "A decade of agile methodologies: Towards explaining agile software development." (2012): 1213-1221.

[2] Serrador, Pedro, and Jeffrey K. Pinto. "Does Agile work?—A quantitative analysis of agile project success." International Journal of Project Management 33.5 (2015): 1040-1051.

[3] Manifesto, Agile. "Manifesto for Agile Software Development, 2001." URL: http://agilemanifesto.org (2010).

[4] Fowler, Martin, and Jim Highsmith. "The agile manifesto." Software Development 9.8 (2001): 28-35.

[5] Ashmore, Sondra, and Kristin Runyan. Introduction to agile methods. Addison-Wesley Professional, 2014.

[6] Rubin, Kenneth S. Essential Scrum: A practical guide to the most popular Agile process. Addison-Wesley, 2012.

[7] Taheri, Mohsen, and Seyed Masoud Sadjadi. "A Feature-Based Tool-Selection Classification for Agile Software Development." SEKE. 2015.

A Comparative Analysis of Classification Algorithms in Diabetic Retinopathy Screening

Saboora Mohammadian, Ali Karsaz
Electrical Engineering Department
Khorasan Institute of Higher Education
Mashhad, Iran
sa.m.roshan@khorasan.ac.ir, karsaz@khorasan.ac.ir

Yaser M. Roshan
Department of Electrical Engineering
Point Park University
Pittsburgh, PA, US
yroshan@pointpark.edu

***Abstract*— Automated screening of diabetic retinopathy plays an important role in diagnosis of the disease in early stages and preventing blindness in patients with diabetes. Various machine learning approaches have been studied in literature with the purpose of improving the accuracy of the screening methods. In this study a comparative analysis of nine common classification algorithms is performed to select the most applicable approach for the specific problem of screening diabetic retinopathy patients.**

Keywords-diabetic retinopathy; machine learning; classification

I. INTRODUCTION

Diabetic Retinopathy (DR) is an eye disease, which is caused by the damage occurs in the retina due to diabetes. Typically, DR is asymptomatic until it becomes a serious threat for vision; therefore, diagnosing DR at early stages can be crucial to increase the chances of early treatment. By automatic screening, DR can be detected in early stages, while minimizing subjectivity and human errors in the manual approach.

II. DATASET AND CLASSIFIERS

A. Diabetic Retinopathy Dataset

The dataset from UCI Diabetic Retinopathy is used in this study [1]. Features of this dataset have been extracted from the publicly available Messidor database of 1151 fundus images of patients [2]. The input data includes 20 attributes of each of the 1151 images, which includes the images features as discussed in [3], as well as the existence of diabetic retinopathy. For training and testing phases of the classifiers, 30 percent and 70 percent of the available data have been selected randomly as the testing and training set, respectively.

B. Selected Classifiers

Selected classifiers are, K-Nearest Neighbors, Decision Tree, Naive Bayes, Random Forest, Adaptive Boosting, Quadratic Discriminant Analysis (QDA), Gaussian Process, Support Vector Machine, and Multi-Layer Perceptron (MLP) Neural Network. All of the classifiers are tested and optimized for DR screening. To measure the performance of the classifiers and as a systematic comparison approach, four typical parameters of accuracy, precision, recall, and F1-score are implemented.

III. RESULTS AND DISCUSSION

Table I demonstrates the comparison between the best performance values for different classification technique. The Gaussian process classifier demonstrates the best results in terms of classifying the diabetic retinopathy cases, while the SVM with polynomial kernel function (of degree 3) and adaptive boosting approach are also showing acceptable performance.

IV. CONCLUSION

The study results demonstrate a qualitative comparison between each classifier and its best tuned parameter. Future studies can leverage these results in terms of pre-selecting the classification algorithm and tuning parameters.

TABLE I. CLASSIFICATION ALGORITHMS PERFORMANCE

Classifier	Optimum Parameter	Performance Indexes Accuracy (percision, recall, F1)
Adaptive Boosting	None	0.8319 (0.8290, 0.8305, 0.8296)
Decision Tree	Maximum depth: 2	0.7316 (0.7573, 0.7477, 0.7297)
Naive-Bayes	None	0.7283 (0.7233, 0.7176, 0.7182)
Gaussian Process	Gain: 4 σ: 1	0.8707 (0.8796, 0.8799, 0.8707)
KNN	Neighbors: 44	0.7607 (0.8074, 0.7802, 0.7574)
QDA	None	0.6247 (0.8709, 0.5672, 0.4141)
Random Forest	Number of trees: 101 Depth: 1100	0.8028 (0.8070, 0.8088, 0.8026)
SVM	Polynomial (d: 3) Penalty: 174.5	0.8449 (0.8651, 0.8585, 0.8447)
MLP	Rectified linear Hidden layers: 61 Penalty: 0.1	0.8125 (0.8709, 0.8190, 0.8124)

REFERENCES

[1] "UCI machine learning repository: Diabetic Retinopathy Debrecen data set data set," 2014. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set#. Accessed: Nov. 10, 2016.

[2] G. Patry, G. Gauthier, B. Lay, J. Roger, and D. Elie, "ADCIS Download Third party: Messidor database," ADCIS S.A., 2016. [Online]. Available: http://messidor.crihan.fr. Accessed: Nov. 16, 2016.

[3] B. Antal and A. Hajdu, "An ensemble-based system for automatic screening of diabetic retinopathy," Knowledge-Based Systems, vol. 60, pp. 20–27, Apr. 2014.

[1] DOI Reference Number 10.18293 / SEKE2017-207

A Conversational Workflow Model for Chatbot

Francesco Colace, Francesco Pascale
DIIn
University of Salerno
Fisciano (SA), Italy
{fcolace, fpascale}@unisa.it

Antonio Ferraioli, Luca Garofalo, Alfredo Troiano
System Management S.r.l.
Naples, Italy
{aferraioli, lgarofalo, atroiano} @sysmanagement.it

Saverio Lemma, Marco Lombardi
SIMASLab
University of Salerno
Fisciano (SA), Italy
{slemma, malombardi} @unisa.it

***Extended Abstract*—**This paper presents the realization of a prototype of a conversational workflow for a Chatbot in tires domain. The initial purpose has focused on the design of the specific model to manage communication and propose the most suitable tires for users. For this aim, it has been used the Petri Net. Finally, after the implementation of the designed model, experimental campaign was conducted in order to demonstrate its enforceability and efficiency. In research fields, the issue of Chatbot and Bot in general has been discussed for many years, although it has seen an increasingly gradual slowdown in recent years. In fact, the amount of investment by companies in trying to create a Bot as similar to an operator is growing[1][2]. The main aim of this paper is to describe a software module prototype, called workflow manager, who is responsible for the management of the flow of conversation between a Chatbot and a user, applied at a real case. Our case study concerns a Chatbot, which will serve for help the choice of tires to buy. After an analysis on the state of art, we have realized the general pattern of operation of the bot and then using the Petri Nets, we have realized the workflow model. Finally, it was conducted the experimental phase that has highlighted the strengths and weaknesses points of the Bot. In the next section, the related works are presented. The design of Workflow Manager was divided in two phases: in the first place, is provided the design of model able to explicitly describe and represent the considered knowledge domain (tires). In the second place, the purpose is to define a workflow module. Therefore, the first aim has been to build an ontology to describe the reference taxonomy. The advantage of this approach is to be able to represent concepts, properties, attributes and constraints of the one part and of the other part a reference model for the workflow[3][4]. In this work, it can be considered the child pneumatic concept. It has several children including the vehicle node, whose descendants are the cars and motorcycles node, and the node size. Each of these nodes presents, in turn, determined descendants. A choice of this type of structure has been effected because, generally, to find a specific pneumatic, must be indicated in detail the characteristics of the vehicle or the size of the eraser. Then, the ontology allows achieving a knowledge and the links between concepts (for example, we can see that we have various types of vehicles or that the concept of car is made up of the attributes of the brand, model, version and year). The next goal was to provide a reasoner that can access in a reference ontology and that it was able to generate and follow a consistent and efficient workflow, defined by a sequence of steps. The conversational module then must be able to navigate autonomously the ontology, moving through the concepts relationships. The selected model is the Petri Net because it lends well for this kind of applications. This model is composed of a set of basic objects: places, transactions and arcs, graphically represented by circles, rectangles and oriented lines. For evaluating the performance of the proposed system an experimental campaign has been developed. An implementation of the Chatbot has been developed and inserted in a Web Site of a tires seller. At the end of the chat session, an email with the suggested model tires has been sent to the potential customer. The email, showed in the store, guaranteeing a 5% discount on tires' price. In this way, it was possible to check whether the tires suggested by the system were right for the car and the needs of the customer. In two months about five hundreds potential customers (identified with the email address) used the Chatbot and 173 of them showed the email in the store and bought tires. The experimental analysis has been conducted about these 173 customers. First of all the performance of the Chatbot in providing the correct suggestions to the user has been evaluated. In particular, three different situations has been considered: Chatbot furnishes a correct suggestion, Chatbot furnishes a correct suggestion, but it does not fit with the real needs of the customer, Chatbot furnishes a wrong suggestion. The obtained results are the following Chatbot furnished the following results: Correct Suggestion: 113 - 65,32%, Correct Suggestion, but not suitable for the needs of the customer: 24 - 13,87%, Wrong Suggestion: 36 - 20,81%. Analyzing the Wrong Suggestion case, we noticed that the system fails when customer talks about a model that have various versions because it proposes tires of different dimensions. Another critical aspect occurs when the system does not understand what kind of vehicle the customer is considering. In the case of Correct Suggestion, but not suitable for the needs of the customer the main problem is in the identification of the real user needs. From the point of view of the usability a questionnaire about his/her interaction with the Chatbot was submitted to each customer. In general, they find the Chatbot easy to use and user friendly. Comparing it with other Chatbot (for example Telegram Chatbot or similar) customers says that our system is more simple and effective. In this paper, an original approach to a Chatbot has been introduced. In particular, the proposed system is based on the Petri Net formalism. A real case has been investigated developing a Chatbot, for a tires' seller. The results obtained by the experimental campaign are satisfying and show the good perspective of this kind of approach. Further developments involve the application of the proposed approach in various contexts and an improvement of the recommender approach.

REFERENCES

[1] M. Casillo, F. Colace, S. Lemma, M. Lombardi, and A. Pietrosanto, "An Ontological Approach to Digital Storytelling," In Proceedings of the The 3rd Multidisciplinary International Social Networks Conference on SocialInformatics 2016, Data Science 2016 (p. 27). ACM, 2016.

[2] F. Colace, L. Greco, S. Lemma, M. Lombardi, D. Yung, and S.K. Chang, "An Adaptive Contextual Recommender System: a Slow Intelligence Perspective," The Twenty-Seventh International Conference on Software Engineering and Knowledge Engineering (SEKE), pp. 64-71, 2015.

[3] F. Colace, M. De Santo, L. Greco, F. Amato, V. Moscato, A. Picariello, "Terminological ontology learning and population using latent dirichlet allocation," Journal of Visual Languages and Computing, 25 (6), pp. 818-826, 2014.

[4] S.K. Chang, D. Yung, F. Colace, L. Greco, S. Lemma, M. Lombardi, "An adaptive contextual recommender system: A slow intelligence perspective," Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE, 2015-January, pp. 64-71, 2015.

DOI reference number: 10.18293/SEKE2017-208

Authors' Index

Program Committee Reviewers' Index

Silvia T. Acuña	Universidad Autonoma de Madrid
Shadi Alawneh	Oakland University
Taisira Albalushi	Sultan Qaboos University
Mark Allison	University of Michigan-Flint
Izzat Alsmadi	Yarmouk University
John Anvik	Department of Mathematics and Computer Science, University of Lethbridge
Doo-Hwan Bae	KAIST
Kyungmin Bae	POSTECH
Ebrahim Bagheri	Ryerson University
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
Bin Cao	College of Computer Science and Technology, Zhejiang University of Technology
Jiannong Cao	Hong Kong Polytechnic University
Nabendu Chaki	University of Calcutta
Chih-Hung Chang	Dept. Of Information Management, Hsiuping University of Science and Technology
Lily Chang	University of Wisconsin, Platteville
Shikuo Chang	Uinversity of Pittsburgh
Meiru Che	The University of Texas at Austin
Wen-Hui Chen	National Taipei University of Technology
Kim-Kwang Raymond Choo	The University of Texas at San Antonio, USA
Stelvio Cimato	Dipartimento di Informatica, Università degli Studi di Milano
Fabio Costa	Federal University of Goias
Jose Luis de La Vara	Carlos III University of Madrid
Peng Di	UNSW
Scott Dick	University of Alberta
Junhua Ding	East Carolina University
Yongfeng Dong	Heibei University of Technology
Zhijiang Dong	Middle Tennessee State University
Derek Doran	Wright State University
Yucong Duan	Hainan University
Philippe Dugerdil	HEG-Univ. of Applied Sciences, Department of Information Systems
Christof Ebert	Vector
Ali Ebnenasir	Michigan Technological University
Magdalini Eirinaki	Computer Engineering Dept, San Jose State University
Abdelrahman Elfaki	University of Tabuk
Mahmoud Elish	Gulf University for Science and Technology
Ruby Elkharboutly	Quinnipiac Univesity
Iaakov Exman	The Jerusalem College of Engineering

Davide Falessi	Cal Poly, USA
Yujian Fu	Alabama A&M University
Honghao Gao	Shanghai University
Kun Gao	zhejiang business technology institute
Felix Garcia	Alarcos Research Group, Information and Systems Department, Escuela Superior de informática, University of Castilla-La Mancha
Ignacio García	University of Castilla-La Mancha
Raúl García-Castro	Universidad Politecnica de Madrid
Wolfgang Golubski	Westsächsische Hochschule Zwickau
Anurag Goswami	North Dakota State University
Christiane Gresse von Wangenheim	Federal University of Santa Catarina - UFSC
Katarina Grolinger	UWO
Hao Han	The University of Tokyo
Xudong He	Florida International University
Robert Heinrich	Karlsruher Institute of Technology
Rubing Huang	Jiangsu University
Yu Huang	Peking University
Bassey 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
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 Goaer	LIUPPA, Université de Pau et des Pays de l'Adour
Yu Lei	University of Texas at Arlington
Ying Li	
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
Kaikai Liu	San Jose State University
Shih-Hsi Liu	California State University, Fresno
Ting Liu	Xi'an Jiaotong Univerisity
Xiaodong Liu	Edinburgh Napier University
Luanna Lopes Lobato	Universidade Federal de Pernambuco - UFPE
Antonella Longo	Wpo, University of Salento
Jian Lu	Nanjing University
Xiangfeng Luo	Computer daparment,Shanghai University,149, Yanchang Road,Shanghai, 200072, China P.R

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	Universidad ORT Uruguay
Hsing Mei	Fu Jen Catholic University
Andre Menolli	Universidade Estadual do Norte do Paraná - UENP
Ali Mili	NJIT
Alok Mishra	Atilim University, Incek 06836, Ankara - Turkey
Hiroyuki Nakagawa	Osaka University
Alex Norta	Department of Informatics, Tallinn University of Technology
Amjad Nusayr	UHV
Edson Oliveirajr	State University of Maringá
Oscar Pereira	University of Aveiro
Angelo Perkusich	Electrical Engineering Department, Federal University of Campina Grande
Antonio Piccinno	University of Bari
Alfonso Pierantonio	University of L'Aquila
Rick Rabiser	Christian Doppler Lab. MEVSS, Johannes Kepler University Linz
Claudia Raibulet	University of Milano-Bicocca
Damith Rajapakse	National University of Singapore
Rajeev Raje	IUPUI
Santanu Rath	NIT Rourkela
Marek Reformat	University of Alberta
Robert Reynolds	Wayne State University
Hassan Reza	University of North Dakota
Ivan Rodero	Rutgers, The State University of New Jersey
Daniel Rodriguez	The University of Alcalá
Samira Sadaoui	University of Regina
Claudio Sant'Anna	Federal University of Bahia
Tanmoy Sarkar	Microsoft Corporation
Klaus-Dieter Schewe	Software Competence Center Hagenberg
Abdelhak Seriai	Lirmm/université Montpellier 2
Michael Shin	Texas Tech University
Martin Solari	Universidad ORT Uruguay
George Spanoudakis	Department of Computer Science, City University
Vijayan Sugumaran	School of Business Administration, Oakland University, Rochester, MI 48309, USA
Jing Sun	The University of Auckland
Meng Sun	Peking University
Yanchun Sun	School of Electronics Engineering and Computer Science, Peking University, China
Gerson Sunyé	Université de Nantes
Chuanqi Tao	Nanjing University of Aeronautics and Astronautics
Mark Trakhtenbrot	Holon Institute of Technology

Hong-Linh Truong	TU Wien, Vienna, Austria
Peter Tröger	TU Chemnitz
T.H. Tse	The University of Hong Kong
Sylvain Vauttier	LG2IP / Ecole des Mines d'Alès
Silvia Vergilio	UFPR
Gennaro Vessio	University of Bari
Sergiy Vilkomir	East Carolina University
Arndt Vonstaa	PUC-Rio
Huanjing Wang	Western Kentucky University
Jue Wang	sccas
Xiaoyin Wang	University of Texas at San Antonio
Ye Wang	Zhejiang Gongshang University
Yong Wang	cqupt
Yong Wang	Texas A&M University
Zhongjie Wang	Harbin Institute of Technology
Ziyuan Wang	Nanjing University of Posts and Telecommunications
Hironori Washizaki	Waseda 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	Hangzhou Dianzi University
Dianxiang Xu	Boise State University
Haiping Xu	University of Massachusetts Dartmouth
Lai Xu	Bournemouth University
Weifeng Xu	Bowie State University
Zheng Xu	Tsinghua University
Guowei Yang	Texas State University
Hongji Yang	Bath Spa University
Jianwei Yin	
Yuyu Yin	HDU
Huiqun Yu	East China University of Science and Technology
Du Zhang	Macau University of Science and Technology
Jilin Zhang	
Pengcheng Zhang	College of Computer and Information Engineering,Hohai University
Shunxiang Zhang	Anhui University of Science & Technology
Yong Zhang	Web and Software Technology R&D center, Tsinghua University, Beijing, P.R.China
Zhigao Zheng	HUST
Hong Zhu	Oxford Brookes University
Huibiao Zhu	East China Normal University
Xingquan Zhu	Florida Atlantic University
Eugenio Zimeo	University of Sannio

External Reviewers' Index

Alam, Moksedul
Allian, Ana Paula
Andrade, Paulo
Ayora, Clara

Baek, Youngmin
Barat, Souvik
Bashari, Mahdi
Bauder, Richard

Cassano, Fabio
Castro, John W.
Chandrasekaran, Jaganmohan
Chen, Yang
Cho, Chiwoo
Chowdhury, Nahida Sultana
Costa Gorgônio, Kyller
Cuzzola, John

Dasgupta, Moitreyee
de Groof, Richard
Deng, Lin
Duan, Feng
Duan, Yucong

Feng, Huadong
Fioravanti, Maria Lydia

Guan, Yue
Guo, Jianmei
Guy, Ed

Ho, Michael

Jee, Eunkyoung
Jiang, Tao

Kuang, Hongyu
Kumar, Arun

Li, Jingwei
Li, Liangqiang
Li, Yi
Liu, Guannan

Liu, Xiao
Liu, Yunhuai
Luo, Xiangfeng

Ma, Yuqin
Magües, Daniel A.

Nirkhe, Vivek
Nistala, Padmalata
Noorian, Mahdi

Paixão, Klérisson
Parodi, Eugenia
Passos Scatalon, Lilian
Perkusich, Mirko
Pimentel, Andrey
Poots, Kent
Prusa, Joseph

Reynolds, Zachary
Rodríguez, Francy
Rostami, Kiana
Roychoudhury, Suman

Shin, Donghwan
Sobrinho, Alvaro
Sobrinho, Élder V.
Song, Jiyoung
Sunkle, Sagar

Tang, Mingdong
Taspolatoglu, Emre
Theis Geraldi, Ricardo
Tibermacine, Chouki

Wang, Luyao
Wang, Yong
Wei, Xiao

Xu, Lihua

Zhang, Junsheng
Zhang, Yinghui
Zhou, Zhangbing
Zhu, Jiaqi
Zhuang, Yan

CALL FOR PAPERS

Journal of Visual Languages and Sentient Systems

http://ksiresearchorg.ipage.com/vlss/

An Open Access Journal published by
KSI Research Inc., 156 Park Square Lane, Pittsburgh, PA 15238 USA
E-mail: vlss@ksiresearch.org

Scope: The Journal of Visual Languages and Sentient Systems (VLSS) is intended to be a forum for researchers, practitioners and developers to exchange ideas and research results, for the advancement of visual languages and sentient multimedia systems. The success of visual languages especially iconic languages is evident to everyone because most smart phones these days use iconic languages to communicate with the end user. Ironically the success of visual languages in practice has led to doubt and uncertainty about the future of visual languages research. However the advances of sentient systems can motivate more research on visual languages, both at the practical level and at the theoretical level.

Sentient systems are distributed systems capable of actively interacting with the environment by gathering, processing, interpreting, storing and retrieving multimedia information originated from sensors, robots, actuators, websites and other information sources. In order for sentient systems to function efficiently and effectively, visual languages may play an important role. To stimulate research towards that goal, the Journal of Visual Languages and Sentient Systems is born.

Types of Papers: VLSS publishes research papers, state-of-the-art surveys, review articles, in the areas of visual languages, sentient multimedia systems, distributed multimedia systems, sensor networks, multimedia interfaces, visual communication, multi-media communications, cognitive aspects of sensor-based systems, and parallel/distributed/neural computing & representations for multimedia information processing. Papers are also welcome to report on actual use, experience, transferred technologies in sentient multimedia systems and applications. Timely research notes, viewpoint articles, book reviews and tool reviews, not to exceed three pages, can also be submitted to VLSS.

Manuscript Preparation: Manuscripts shall be submitted electronically to VLSS. Original papers only will be considered. Manuscripts should follow the double-column format and be submitted in the form of a pdf file. Page 1 should contain the article title, author(s), and affiliation(s); the name and complete mailing address of the person to whom correspondence should be sent, and a short abstract (100-150 words). Any footnotes to the title (indicated by *, +, etc.) should be placed at the bottom of page 1.

Manuscripts are accepted for review with the understanding that the same work has not been and will not be nor is presently submitted elsewhere, and that its submission for publication has been approved by all of the authors and by the institution where the work was carried out; further, that any person cited as a course of personal communications has approved such citation. Written authorization may be required at the Editor's discretion. Articles and any other material published in VLSS represent the opinions of the author(s) and should not be construed to reflect the opinions of the Editor(s) and the Publisher.

How to Submit: Paper submission should be through https://www.easychair.org/conferences/?conf=dmsvlss2017. First you need to join EasyChair as an author. Then you select the paper type "Journal of Visual Languages and Sentient Systems", or the paper type “VLSS SI: special issue title” if you are submitting to a VLSS Special Issue.

DOI: Every paper accepted by VLSS will be assigned a unique DOI reference number.

For More Information: Contact vlss@ksiresearch.org

KSI Research Inc. 156 Park Square Lane, Pittsburgh, PA 15238 USA Tel: 412-606-5022