

Beniamino Murgante Osvaldo Gervasi
Sanjay Misra Nadia Nedjah
Ana Maria A.C. Rocha David Taniar
Bernady O. Apduhan (Eds.)

LNCS 7336

Computational Science and Its Applications – ICCSA 2012

12th International Conference
Salvador de Bahia, Brazil, June 2012
Proceedings, Part IV

4
Part IV

 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Beniamino Murgante Osvaldo Gervasi
Sanjay Misra Nadia Nedjah
Ana Maria A.C. Rocha David Taniar
Bernady O. Apduhan (Eds.)

Computational Science and Its Applications – ICCSA 2012

12th International Conference
Salvador de Bahia, Brazil, June 18-21, 2012
Proceedings, Part IV

Volume Editors

Beniamino Murgante

University of Basilicata, Potenza, Italy, E-mail: beniamino.murgante@unibas.it

Oswaldo Gervasi

University of Perugia, Italy, E-mail: osvaldo@unipg.it

Sanjay Misra

Federal University of Technology, Minna, Nigeria, E-mail: smisra@futminna.edu.ng

Nadia Nedjah

State University of Rio de Janeiro, Brazil, E-mail: nadia@eng.uerj.br

Ana Maria A. C. Rocha

University of Minho, Braga, Portugal, E-mail: arocha@dps.uminho.pt

David Taniar

Monash University, Clayton, VIC, Australia, E-mail: david.taniar@infotech.monash.edu.au

Bernady O. Apduhan

Kyushu Sangyo University, Fukuoka, Japan, E-mail: bob@is.kyusan-u.ac.jp

ISSN 0302-9743

e-ISSN 1611-3349

ISBN 978-3-642-31127-7

e-ISBN 978-3-642-31128-4

DOI 10.1007/978-3-642-31128-4

Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012939389

CR Subject Classification (1998): C.2.4, C.2, H.4, F.2, H.3, D.2, F.1, H.5, H.2.8, K.6.5, I.3

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

© Springer-Verlag Berlin Heidelberg 2012

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This four-part volume (LNCS 7333-7336) contains a collection of research papers from the 12th International Conference on Computational Science and Its Applications (ICCSA 2012) held in Salvador de Bahia, Brazil, during June 18–21, 2012. ICCSA is one of the successful international conferences in the field of computational sciences, and this year for the first time in the history of the ICCSA conference series it was held in South America. Previously the ICCSA conference series have been held in Santander, Spain (2011), Fukuoka, Japan (2010), Suwon, Korea (2009), Perugia, Italy (2008), Kuala Lumpur, Malaysia (2007), Glasgow, UK (2006), Singapore (2005), Assisi, Italy (2004), Montreal, Canada (2003), (as ICCS) Amsterdam, The Netherlands (2002), and San Francisco, USA (2001).

The computational science community has enthusiastically embraced the successive editions of ICCSA, thus contributing to making ICCSA a focal meeting point for those interested in innovative, cutting-edge research about the latest and most exciting developments in the field. We are grateful to all those who have contributed to the ICCSA conference series.

ICCSA 2012 would not have been made possible without the valuable contribution of many people. We would like to thank all session organizers for their diligent work, which further enhanced the conference level, and all reviewers for their expertise and generous effort, which led to a very high quality event with excellent papers and presentations. We specially recognize the contribution of the Program Committee and local Organizing Committee members for their tremendous support and for making this congress a very successful event. We would like to sincerely thank our keynote speakers, who willingly accepted our invitation and shared their expertise.

We also thank our publisher, Springer, for accepting to publish the proceedings and for their kind assistance and cooperation during the editing process.

Finally, we thank all authors for their submissions and all conference attendees for making ICCSA 2012 truly an excellent forum on computational science, facilitating the exchange of ideas, fostering new collaborations and shaping the future of this exciting field. Last, but certainly not least, we wish to thank our readers for their interest in this volume. We really hope you find in these pages interesting material and fruitful ideas for your future work.

We cordially invite you to visit the ICCSA website—<http://www.iccsa.org>—where you can find relevant information about this interesting and exciting event.

June 2012

Oswaldo Gervasi
David Taniar

Organization

ICCSA 2012 was organized by Universidade Federal da Bahia (Brazil), Universidade Federal do Recôncavo da Bahia (Brazil), Universidade Estadual de Feira de Santana (Brazil), University of Perugia (Italy), University of Basilicata (Italy), Monash University (Australia), and Kyushu Sangyo University (Japan).

Honorary General Chairs

Antonio Laganà	University of Perugia, Italy
Norio Shiratori	Tohoku University, Japan
Kenneth C.J. Tan	Qontix, UK

General Chairs

Oswaldo Gervasi	University of Perugia, Italy
David Taniar	Monash University, Australia

Program Committee Chairs

Bernady O. Apduhan	Kyushu Sangyo University, Japan
Beniamino Murgante	University of Basilicata, Italy

Workshop and Session Organizing Chairs

Beniamino Murgante	University of Basilicata, Italy
--------------------	---------------------------------

Local Organizing Committee

Frederico V. Prudente	Universidade Federal da Bahia, Brazil (Chair)
Mirco Ragni	Universidade Estadual de Feira de Santana, Brazil
Ana Carla P. Bitencourt	Universidade Federal do Recôncavo da Bahia, Brazil
Cassio Pigozzo	Universidade Federal da Bahia, Brazil
Angelo Duarde	Universidade Estadual de Feira de Santana, Brazil
Marcos E. Barreto	Universidade Federal da Bahia, Brazil
José Garcia V. Miranda	Universidade Federal da Bahia, Brazil

International Liaison Chairs

Jemal Abawajy	Deakin University, Australia
Marina L. Gavrilova	University of Calgary, Canada
Robert C.H. Hsu	Chung Hua University, Taiwan
Tai-Hoon Kim	Hannam University, Korea
Andrés Iglesias	University of Cantabria, Spain
Takashi Naka	Kyushu Sangyo University, Japan
Rafael D.C. Santos	National Institute for Space Research, Brazil

Workshop Organizers

Advances in High-Performance Algorithms and Applications (AHPAA 2012)

Massimo Cafaro	University of Salento, Italy
Giovanni Aloisio	University of Salento, Italy

Advances in Web-Based Learning (AWBL 2012)

Mustafa Murat Inceoglu	Ege University, Turkey
------------------------	------------------------

Bio-inspired Computing and Applications (BIOCA 2012)

Nadia Nedjah	State University of Rio de Janeiro, Brazil
Luiza de Macedo Mourell	State University of Rio de Janeiro, Brazil

Computer-Aided Modeling, Simulation, and Analysis (CAMSA 2012)

Jie Shen	University of Michigan, USA
Yuqing Song	Tianjing University of Technology and Education, China

Cloud Computing and Its Applications (CCA 2012)

Jemal Abawajy	University of Deakin, Australia
Osvaldo Gervasi	University of Perugia, Italy

Computational Geometry and Applications (CGA 2012)

Marina L. Gavrilova	University of Calgary, Canada
---------------------	-------------------------------

Chemistry and Materials Sciences and Technologies (CMST 2012)

Antonio Laganà University of Perugia, Italy

Cities, Technologies and Planning (CTP 2012)

Giuseppe Borruso University of Trieste, Italy
Beniamino Murgante University of Basilicata, Italy

Computational Tools and Techniques for Citizen Science and Scientific Outreach (CTTCS 2012)

Rafael Santos National Institute for Space Research, Brazil
Jordan Raddick and Johns Hopkins University, USA
Ani Thakar Johns Hopkins University, USA

Econometrics and Multidimensional Evaluation in the Urban Environment (EMEUE 2012)

Carmelo M. Torre Polytechnic of Bari, Italy
Maria Cerreta Università Federico II of Naples, Italy
Paola Perchinunno University of Bari, Italy

Future Information System Technologies and Applications (FISTA 2012)

Bernady O. Apduhan Kyushu Sangyo University, Japan

Geographical Analysis, Urban Modeling, Spatial Statistics (GEOG-AN-MOD 2012)

Stefania Bertazzon University of Calgary, Canada
Giuseppe Borruso University of Trieste, Italy
Beniamino Murgante University of Basilicata, Italy

International Workshop on Biomathematics, Bioinformatics and Biostatistics (IBBB 2012)

Unal Ufuktepe Izmir University of Economics, Turkey
Andrés Iglesias University of Cantabria, Spain

International Workshop on Collective Evolutionary Systems (IWCES 2012)

Alfredo Milani
Clement Leung

University of Perugia, Italy
Hong Kong Baptist University, Hong Kong

Mobile Communications (MC 2012)

Hyunseung Choo

Sungkyunkwan University, Korea

Mobile Computing, Sensing, and Actuation for Cyber Physical Systems (MSA4CPS 2012)

Moonseong Kim
Saad Qaisar

Korean intellectual Property Office, Korea
NUST School of Electrical Engineering and
Computer Science, Pakistan

Optimization Techniques and Applications (OTA 2012)

Ana Maria Rocha

University of Minho, Portugal

Parallel and Mobile Computing in Future Networks (PMCFUN 2012)

Al-Sakib Khan Pathan

International Islamic University Malaysia,
Malaysia

PULSES - Transitions and Nonlinear Phenomena (PULSES 2012)

Carlo Cattani
Ming Li
Shengyong Chen

University of Salerno, Italy
East China Normal University, China
Zhejiang University of Technology, China

Quantum Mechanics: Computational Strategies and Applications (QMCSA 2012)

Mirco Ragni
Frederico Vasconcellos

Universidade Federal de Bahia, Brazil

Prudente
Angelo Marconi Maniero
Ana Carla Peixoto Bitencourt

Universidade Federal de Bahia, Brazil
Universidade Federal de Bahia, Brazil
Universidade Federal do Recôncavo da Bahia,
Brazil

Remote Sensing Data Analysis, Modeling, Interpretation and Applications: From a Global View to a Local Analysis (RS 2012)

Rosa Lasaponara Institute of Methodologies for Environmental
Analysis, National Research Council, Italy
Nicola Masini Archaeological and Monumental Heritage
Institute, National Research Council, Italy

Soft Computing and Data Engineering (SCDE 2012)

Mustafa Matt Deris Universiti Tun Hussein Onn Malaysia, Malaysia
Tutut Herawan Universitas Ahmad Dahlan, Indonesia

Software Engineering Processes and Applications (SEPA 2012)

Sanjay Misra Federal University of Technology Minna,
Nigeria

Software Quality (SQ 2012)

Sanjay Misra Federal University of Technology Minna,
Nigeria

Security and Privacy in Computational Sciences (SPCS 2012)

Arijit Ukil Tata Consultancy Services, India

Tools and Techniques in Software Development Processes (TTSDP 2012)

Sanjay Misra Federal University of Technology Minna,
Nigeria

Virtual Reality and Its Applications (VRA 2012)

Oswaldo Gervasi University of Perugia, Italy
Andr es Iglesias University of Cantabria, Spain

Wireless and Ad-Hoc Networking (WADNet 2012)

Jongchan Lee
Sangjoon Park

Kunsan National University, Korea
Kunsan National University, Korea

Program Committee

Jemal Abawajy	Daekin University, Australia
Kenny Adamson	University of Ulster, UK
Filipe Alvelos	University of Minho, Portugal
Paula Amaral	Universidade Nova de Lisboa, Portugal
Hartmut Asche	University of Potsdam, Germany
Md. Abul Kalam Azad	University of Minho, Portugal
Michela Bertolotto	University College Dublin, Ireland
Sandro Bimonte	CEMAGREF, TSCF, France
Rod Blais	University of Calgary, Canada
Ivan Bleic	University of Sassari, Italy
Giuseppe Borruso	University of Trieste, Italy
Alfredo Buttari	CNRS-IRIT, France
Yves Caniou	Lyon University, France
José A. Cardoso e Cunha	Universidade Nova de Lisboa, Portugal
Leocadio G. Casado	University of Almeria, Spain
Carlo Cattani	University of Salerno, Italy
Mete Celik	Erciyes University, Turkey
Alexander Chemeris	National Technical University of Ukraine “KPI”, Ukraine
Min Young Chung	Sungkyunkwan University, Korea
Gilberto Corso Pereira	Federal University of Bahia, Brazil
M. Fernanda Costa	University of Minho, Portugal
Gaspar Cunha	University of Minho, Portugal
Carla Dal Sasso Freitas	Universidade Federal do Rio Grande do Sul, Brazil
Pradesh Debba	The Council for Scientific and Industrial Research (CSIR), South Africa
Frank Devai	London South Bank University, UK
Rodolphe Devillers	Memorial University of Newfoundland, Canada
Prabu Dorairaj	NetApp, India/USA
M. Irene Falcao	University of Minho, Portugal
Cherry Liu Fang	U.S. DOE Ames Laboratory, USA
Edite M.G.P. Fernandes	University of Minho, Portugal
Jose-Jesus Fernandez	National Centre for Biotechnology, CSIS, Spain
Maria Antonia Forjaz	University of Minho, Portugal
Maria Celia Furtado Rocha	PRODEB–PósCultura/UFBA, Brazil
Akemi Galvez	University of Cantabria, Spain
Paulino Jose Garcia Nieto	University of Oviedo, Spain
Marina Gavrilova	University of Calgary, Canada

Jerome Gensel	LSR-IMAG, France
Maria Giaoutzi	National Technical University, Athens, Greece
Andrzej M. Goscinski	Deakin University, Australia
Alex Hagen-Zanker	University of Cambridge, UK
Malgorzata Hanzl	Technical University of Lodz, Poland
Shanmugasundaram Hariharan	B.S. Abdur Rahman University, India
Eligius M.T. Hendrix	University of Malaga/Wageningen University, Spain/The Netherlands
Hisamoto Hiyoshi	Gunma University, Japan
Fermin Huarte	University of Barcelona, Spain
Andres Iglesias	University of Cantabria, Spain
Mustafa Inceoglu	EGE University, Turkey
Peter Jimack	University of Leeds, UK
Qun Jin	Waseda University, Japan
Farid Karimipour	Vienna University of Technology, Austria
Baris Kazar	Oracle Corp., USA
DongSeong Kim	University of Canterbury, New Zealand
Taihoon Kim	Hannam University, Korea
Ivana Kolingerova	University of West Bohemia, Czech Republic
Dieter Kranzlmüller	LMU and LRZ Munich, Germany
Antonio Laganà	University of Perugia, Italy
Rosa Lasaponara	National Research Council, Italy
Maurizio Lazzari	National Research Council, Italy
Cheng Siong Lee	Monash University, Australia
Sangyoun Lee	Yonsei University, Korea
Jongchan Lee	Kunsan National University, Korea
Clement Leung	Hong Kong Baptist University, Hong Kong
Chendong Li	University of Connecticut, USA
Gang Li	Deakin University, Australia
Ming Li	East China Normal University, China
Fang Liu	AMES Laboratories, USA
Xin Liu	University of Calgary, Canada
Savino Longo	University of Bari, Italy
Tinghuai Ma	NanJing University of Information Science and Technology, China
Sergio Maffioletti	University of Zurich, Switzerland
Ernesto Marcheggiani	Katholieke Universiteit Leuven, Belgium
Antonino Marvuglia	Research Centre Henri Tudor, Luxembourg
Nicola Masini	National Research Council, Italy
Nirvana Meratnia	University of Twente, The Netherlands
Alfredo Milani	University of Perugia, Italy
Sanjay Misra	Federal University of Technology Minna, Nigeria
Giuseppe Modica	University of Reggio Calabria, Italy

José Luis Montaña	University of Cantabria, Spain
Beniamino Murgante	University of Basilicata, Italy
Jiri Nedoma	Academy of Sciences of the Czech Republic, Czech Republic
Laszlo Neumann	University of Girona, Spain
Kok-Leong Ong	Deakin University, Australia
Belen Palop	Universidad de Valladolid, Spain
Marcin Paprzycki	Polish Academy of Sciences, Poland
Eric Pardede	La Trobe University, Australia
Kwangjin Park	Wonkwang University, Korea
Ana Isabel Pereira	Polytechnic Institute of Braganca, Portugal
Maurizio Pollino	Italian National Agency for New Technologies, Energy and Sustainable Economic Development, Italy
Alenka Poplin	University of Hamburg, Germany
Vidyasagar Potdar	Curtin University of Technology, Australia
David C. Prosperi	Florida Atlantic University, USA
Wenny Rahayu	La Trobe University, Australia
Jerzy Respondek	Silesian University of Technology Poland
Ana Maria A.C. Rocha	University of Minho, Portugal
Humberto Rocha	INESC-Coimbra, Portugal
Alexey Rodionov	Institute of Computational Mathematics and Mathematical Geophysics, Russia
Cristina S. Rodrigues	University of Minho, Portugal
Octavio Roncero	CSIC, Spain
Maytham Safar	Kuwait University, Kuwait
Haiduke Sarafian	The Pennsylvania State University, USA
Qi Shi	Liverpool John Moores University, UK
Dale Shires	U.S. Army Research Laboratory, USA
Takuo Suganuma	Tohoku University, Japan
Ana Paula Teixeira	University of Tras-os-Montes and Alto Douro, Portugal
Senhorinha Teixeira	University of Minho, Portugal
Parimala Thulasiraman	University of Manitoba, Canada
Carmelo Torre	Polytechnic of Bari, Italy
Javier Martinez Torres	Centro Universitario de la Defensa Zaragoza, Spain
Giuseppe A. Trunfio	University of Sassari, Italy
Unal Ufuktepe	Izmir University of Economics, Turkey
Mario Valle	Swiss National Supercomputing Centre, Switzerland
Pablo Vanegas	University of Cuenca, Ecuador
Piero Giorgio Verdini	INFN Pisa and CERN, Italy
Marco Vizzari	University of Perugia, Italy
Koichi Wada	University of Tsukuba, Japan

Krzysztof Walkowiak
 Robert Weibel
 Roland Wismüller
 Mudasser Wyne
 Chung-Huang Yang
 Xin-She Yang
 Salim Zabir
 Albert Y. Zomaya

Wroclaw University of Technology, Poland
 University of Zurich, Switzerland
 Universität Siegen, Germany
 SOET National University, USA
 National Kaohsiung Normal University, Taiwan
 National Physical Laboratory, UK
 France Telecom Japan Co., Japan
 University of Sydney, Australia

Sponsoring Organizations

ICCSA 2012 would not have been possible without tremendous support of many organizations and institutions, for which all organizers and participants of ICCSA 2012 express their sincere gratitude:



Universidade Federal da Bahia, Brazil
 (<http://www.ufba.br>)



Universidade Federal do Recôncavo da Bahia,
 Brazil
 (<http://www.ufrb.edu.br>)



Universidade Estadual de Feira de Santana,
 Brazil
 (<http://www.uefs.br>)



University of Perugia, Italy
 (<http://www.unipg.it>)



University of Basilicata, Italy
 (<http://www.unibas.it>)

XVI Organization



MONASH University

Monash University, Australia
(<http://monash.edu>)



九州産業大学
KYUSHU SANGYO UNIVERSITY

Kyushu Sangyo University, Japan
(www.kyusan-u.ac.jp)



Brazilian Computer Society
(www.sbc.org.br)



Coordenação de Aperfeiçoamento de Pessoal de
Nível Superior (CAPES), Brazil
(<http://www.capes.gov.br>)



National Council for Scientific and
Technological Development (CNPq), Brazil
(<http://www.cnpq.br>)



SECRETARIA DE CIÊNCIA,
TECNOLOGIA E INOVAÇÃO



Fundação de Amparo à Pesquisa do Estado
da Bahia (FAPESB), Brazil
(<http://www.fapesb.ba.gov.br>)

Table of Contents – Part IV

Workshop on Software Engineering Processes and Applications (SEPA 2012)

Modeling Road Traffic Signals Control Using UML and the MARTE Profile	1
<i>Eduardo Augusto Silvestre and Michel dos Santos Soares</i>	
Analysis of Techniques for Documenting User Requirements	16
<i>Michel dos Santos Soares and Daniel Souza Cioquetta</i>	
Predicting Web Service Maintainability via Object-Oriented Metrics: A Statistics-Based Approach	29
<i>José Luis Ordiales Coscia, Marco Crasso, Cristian Mateos, Alejandro Zunino, and Sanjay Misra</i>	
Early Automated Verification of Tool Chain Design	40
<i>Matthias Biehl</i>	
Using UML Stereotypes to Support the Requirement Engineering: A Case Study	51
<i>Vitor A. Batista, Daniela C.C. Peixoto, Wilson Pádua, and Clarindo Isaías P.S. Pádua</i>	
Identifying Business Rules to Legacy Systems Reengineering Based on BPM and SOA	67
<i>Gleison S. do Nascimento, Cirano Iochpe, Lucinéia Thom, André C. Kalsing, and Álvaro Moreira</i>	
Abstraction Analysis and Certified Flow and Context Sensitive Points-to Relation for Distributed Programs	83
<i>Mohamed A. El-Zawawy</i>	
An Approach to Measure Understandability of Extended UML Based on Metamodel	100
<i>Yan Zhang, Yi Liu, Zhiyi Ma, Xuying Zhao, Xiaokun Zhang, and Tian Zhang</i>	
Dealing with Dependencies among Functional and Non-functional Requirements for Impact Analysis in Web Engineering	116
<i>José Alfonso Aguilar, Irene Garrigós, Jose-Norberto Mazón, and Anibal Zaldívar</i>	

Assessing Maintainability Metrics in Software Architectures Using COSMIC and UML	132
<i>Eudisley Gomes dos Anjos, Ruan Delgado Gomes, and Mário Zenha-Rela</i>	
Plagiarism Detection in Software Using Efficient String Matching	147
<i>Kusum Lata Pandey, Suneeta Agarwal, Sanjay Misra, and Rajesh Prasad</i>	
Dynamic Software Maintenance Effort Estimation Modeling Using Neural Network, Rule Engine and Multi-regression Approach	157
<i>Ruchi Shukla, Mukul Shukla, A.K. Misra, T. Marwala, and W.A. Clarke</i>	

Workshop on Software Quality (SQ 2012)

New Measures for Maintaining the Quality of Databases	170
<i>Hendrik Decker</i>	
A New Way to Determine External Quality of ERP Software	186
<i>Ali Orhan Aydin</i>	
Towards a Catalog of Spreadsheet Smells	202
<i>Jácome Cunha, João P. Fernandes, Hugo Ribeiro, and João Saraiva</i>	
Program and Aspect Metrics for MATLAB	217
<i>Pedro Martins, Paulo Lopes, João P. Fernandes, João Saraiva, and João M.P. Cardoso</i>	
A Suite of Cognitive Complexity Metrics	234
<i>Sanjay Misra, Murat Koyuncu, Marco Crasso, Cristian Mateos, and Alejandro Zunino</i>	
Complexity Metrics for Cascading Style Sheets	248
<i>Adewole Adewumi, Sanjay Misra, and Nicholas Ikhu-Omoregbe</i>	
A Systematic Review on the Impact of CK Metrics on the Functional Correctness of Object-Oriented Classes	258
<i>Yasser A. Khan, Mahmoud O. Elish, and Mohamed El-Attar</i>	

Workshop on Security and Privacy in Computational Sciences (SPCS 2012)

Pinpointing Malicious Activities through Network and System-Level Malware Execution Behavior	274
<i>André Ricardo Abed Grégio, Vitor Monte Afonso, Dario Simões Fernandes Filho, Paulo Lício de Geus, Mario Jino, and Rafael Duarte Coelho dos Santos</i>	

A Malware Detection System Inspired on the Human Immune System	286
<i>Isabela Liane de Oliveira, André Ricardo Abed Grégio, and Adriano Mauro Cansian</i>	
Interactive, Visual-Aided Tools to Analyze Malware Behavior	302
<i>André Ricardo Abed Grégio, Alexandre Or Cansian Baruque, Vitor Monte Afonso, Dario Simões Fernandes Filho, Paulo Lício de Geus, Mario Jino, and Rafael Duarte Coelho dos Santos</i>	
Interactive Analysis of Computer Scenarios through Parallel Coordinates Graphics	314
<i>Gabriel D. Cavalcante, Sebastien Tricaud, Cleber P. Souza, and Paulo Lício de Geus</i>	
Methodology for Detection and Restraint of P2P Applications in the Network	326
<i>Rodrigo M.P. Silva and Ronaldo M. Salles</i>	
Workshop on Soft Computing and Data Engineering (SCDE 2012)	
Text Categorization Based on Fuzzy Soft Set Theory	340
<i>Bana Handaga and Mustafa Mat Deris</i>	
Cluster Size Determination Using JPEG Files	353
<i>Nurul Azma Abdullah, Rosziati Ibrahim, and Kamaruddin Malik Mohamad</i>	
Semantic Web Search Engine Using Ontology, Clustering and Personalization Techniques	364
<i>Noryusliza Abdullah and Rosziati Ibrahim</i>	
Granules of Words to Represent Text: An Approach Based on Fuzzy Relations and Spectral Clustering	379
<i>Patrícia F. Castro and Geraldo B. Xexéo</i>	
Multivariate Time Series Classification by Combining Trend-Based and Value-Based Approximations	392
<i>Bilal Esmael, Arghad Arnaout, Rudolf K. Fruhwirth, and Gerhard Thonhauser</i>	

General Track on High Performance Computing and Networks

Impact of <i>pay-as-you-go</i> Cloud Platforms on Software Pricing and Development: A Review and Case Study	404
<i>Fernando Pires Barbosa and Andrea Schwertner Charão</i>	
Resilience for Collaborative Applications on Clouds: Fault-Tolerance for Distributed HPC Applications	418
<i>Toàn Nguyễn and Jean-Antoine Désidéri</i>	
T-DMB Receiver Model for Emergency Alert Service	434
<i>Seong-Geun Kwon, Suk-Hwan Lee, Eung-Joo Lee, and Ki-Ryong Kwon</i>	
A Framework for Context-Aware Systems in Mobile Devices	444
<i>Eduardo Jorge, Matheus Farias, Rafael Carmo, and Wesley Vieira</i>	
A Simulation Framework for Scheduling Performance Evaluation on CPU-GPU Heterogeneous System	457
<i>Flavio Vella, Igor Neri, Osvaldo Gervasi, and Sergio Tasso</i>	
Influence of Topology on Mobility and Transmission Capacity of Human-Based DTNs	470
<i>Danilo A. Moschetto, Douglas O. Freitas, Lourdes P.P. Poma, Ricardo Aparecido Perez de Almeida, and Cesar A.C. Marcondes</i>	
Towards a Computer Assisted Approach for Migrating Legacy Systems to SOA	484
<i>Gonzalo Salvatierra, Cristian Mateos, Marco Crasso, and Alejandro Zunino</i>	
1+1 Protection of Overlay Distributed Computing Systems: Modeling and Optimization	498
<i>Krzysztof Walkowiak and Jacek Rak</i>	
Scheduling and Capacity Design in Overlay Computing Systems	514
<i>Krzysztof Walkowiak, Andrzej Kasprzak, Michał Kosowski, and Marek Miziołek</i>	
GPU Acceleration of the <i>caffa3d.MB</i> Model	530
<i>Pablo Igounet, Pablo Alfaro, Gabriel Usera, and Pablo Ezzatti</i>	
Security-Effective Fast Authentication Mechanism for Network Mobility in Proxy Mobile IPv6 Networks	543
<i>Illkyun Im, Young-Hwa Cho, Jae-Young Choi, and Jongpil Jeong</i>	
An Architecture for Service Integration and Unified Communication in Mobile Computing	560
<i>Ricardo Aparecido Perez de Almeida and Hélio C. Guardia</i>	

Task Allocation in Mesh Structure: 2Side LeapFrog Algorithm and Q-Learning Based Algorithm	576
<i>Iwona Pozniak-Koszalka, Wojciech Proma, Leszek Koszalka, Maciej Pol, and Andrzej Kasprzak</i>	
Follow-Us: A Distributed Ubiquitous Healthcare System Simulated by MannaSim	588
<i>Maria Luísa Amarante Ghizoni, Aduino Santos, and Linnyer Beatrys Ruiz</i>	
Adaptive Dynamic Frequency Scaling for Thermal-Aware 3D Multi-core Processors	602
<i>Hong Jun Choi, Young Jin Park, Hsien-Hsin Lee, and Cheol Hong Kim</i>	
A Context-Aware Service Model Based on the OSGi Framework for u-Agricultural Environments	613
<i>Jongsun Choi, Sangjoon Park, Jongchan Lee, and Yongyun Cho</i>	
A Security Framework for Blocking New Types of Internet Worms in Ubiquitous Computing Environments	622
<i>Iksu Kim and Yongyun Cho</i>	
Quality Factors in Development Best Practices for Mobile Applications	632
<i>Euler Horta Marinho and Rodolfo Ferreira Resende</i>	
ShadowNet: An Active Defense Infrastructure for Insider Cyber Attack Prevention	646
<i>Xiaohui Cui, Wade Gasior, Justin Beaver, and Jim Treadwell</i>	
Author Index	655

Modeling Road Traffic Signals Control Using UML and the MARTE Profile

Eduardo Augusto Silvestre and Michel dos Santos Soares

Federal University of Uberlândia (UFU), Computing Faculty (FACOM),
Av. João Naves de Ávila, 2121, Bloco 1B
Uberlândia - Brazil
eduardosilvestre@iftm.edu.br, mics.soares@gmail.com

Abstract. The problem of software modeling and design of road traffic signals control has long been taken into consideration. A variety of modeling languages have been applied in this field. However, still no single modeling language can be considered a standard to model distributed real-time systems such as traffic signals systems. Thus, further evaluation is necessary. In this article, a UML profile created for designing real-time systems, MARTE, is applied to model a traffic signals control system. MARTE is compared with UML and SPT, a former UML profile. The result is that with MARTE, UML models are more specific, but also more complex.

Keywords: MARTE, UML, Road Traffic Control, Real-Time Systems.

1 Introduction

When designing distributed real-time systems for critical infrastructures, such as road traffic, system complexity is increased due to the large number of elements, strict real-time constraints, and reliability factors, as these systems involve human life. There are many characteristics that make the development of distributed real-time systems difficult. Normally, these systems present high complexity, being hard to comprehend, design, implement and verify.

Traffic signals are one of the main approaches to control intersections. They regulate, warn and guide transportation with the purpose of improving safety and efficiency of pedestrians and vehicles. When not well-designed, traffic signals may lead to excessive delays when cycle lengths are too long and increase the risk of collisions.

The behavior of a traffic signal can be modeled as a Discrete Event System (DES). These systems are often large, distributed systems in which events occur at specific instants of time [9]. From a DES point of view, a road junction (intersection) can be seen as a resource that is shared by vehicles at certain instants of time. The design of the control logic of a traffic signal must take care of efficiency and speed, but also of safety and security. The main purpose of traffic signals is to provide safe, efficient and fair crossing of the junction.

The problem of software modeling and design of road traffic signals control has long been taken into consideration, with a variety of modeling languages applied in this field, including Fuzzy Logic [40], Statecharts [14] [16], and Petri nets [21] [33]. Therefore, there is no standard modeling language in this domain.

Since its introduction, UML [29] has been applied to model real-time systems in a variety of domains [11]. Considering the road traffic domain, which is the one of interest of this article (Section 3), UML has been applied in previous works [3] [19]. In many of these works, UML presented flaws that were well-documented. For instance, the representation of time is considered poor, too informal and insufficient, and it does not include deadlocks and periods [6] [7] [13]. Behavior diagrams, such as the Sequence diagram, cannot represent time constraints effectively [34], as they are essentially untimed, expressing only chronological order [2]. UML does not provide mechanisms to describe aspects of task management such as priorities [6].

In order to solve these issues, the SPT profile was proposed [23]. Although the SPT profile was studied and applied in some works [39] [1] [38] [5], it was not well-received by the real-time community [28], mainly because it lacks characteristics to represent hardware platforms, the representation of time is insufficient, and it was not in conformance with UML 2.x. Therefore, a new UML profile for real-time systems, including distributed and embedded systems, was proposed. An overview of the MARTE profile is presented in section 2. The practical application of MARTE is still very incipient [10], but some results have been published in domains such as system-on-chip [31], networks [12] and the development of software product lines [4].

To the best of our knowledge MARTE has not yet been applied to model traffic signals control, as is done in this article through the case study presented in Section 4. In addition, a comparison of MARTE with the former UML profile for real time systems, SPT, and with the pure UML specification is presented in the discussion (Section 5). The comparison is based on our own experience on modeling with MARTE, as shown in this article, but mainly considering articles previously published.

2 Overview on the MARTE Modeling Language

The UML profile MARTE (*Modeling and Analysis of Real-Time and Embedded systems*) [27] is a recently adopted OMG standard that specializes UML by adding concepts for modeling and analysis of real-time and embedded systems. MARTE is an evolution over the former UML profile SPT (*Schedulability, Performance, and Time*) [23].

As illustrated in Figure 1, MARTE is organized around three main packages. The MARTE Foundations package defines concepts for real-time and embedded systems. These concepts cover the modeling of generic applications and platform artifacts. The Foundations package is composed of the following sub-packages: Core Elements (CoreElements), Non-Functional Properties Modeling (NFPs), Time Modeling (Time), Generic Resource Modeling (GRM) and

Allocation Modeling (Alloc). These foundation concepts are refined for design purpose into the MARTE Design Model package and for analysis purpose into the MARTE Analysis Model package [22].

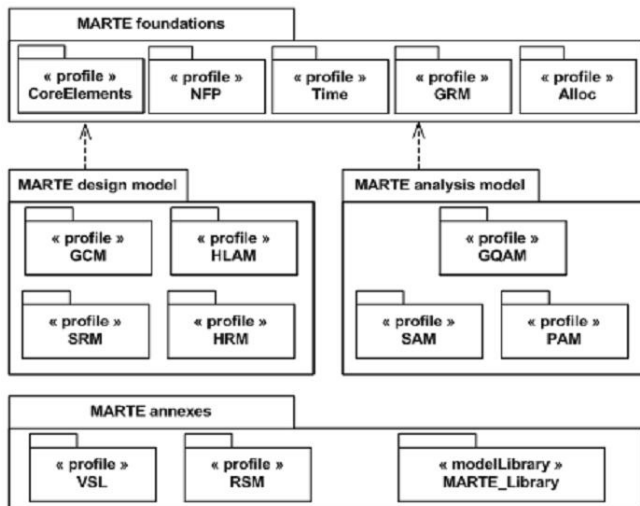


Fig. 1. The MARTE Profile [28]

The MARTE Design Model package provides support required for a variety of activities, from specification to detailed design of real-time systems. It is composed of four sub-packages. The Generic Component Model (GCM) package presents additional concepts to address the modeling of artifacts in the context of real-time systems. The GCM extends the UML component model by adding two specializations of ports: message ports and flow ports.

The High-Level Application Modeling (HLAM) package provides high-level modeling concepts to deal with real-time and embedded features modeling. The HLAM package allows to specify temporal properties of calls. HLAM can model active objects (active components), i.e., entities that provide their own thread of control. The Detailed Resource Modeling (DRM) package provides a set of detailed resources for modeling both software and hardware platforms by specializing the concepts defined within the GRM. DRM is divided into two sub-packages: Software Resource Modeling (SRM) and Hardware Resource Modeling (HRM).

The MARTE Analysis Model package offers concepts for the analysis of models. The analysis of models can detect problems early in the development life cycle and is useful to reduce cost and risk of software development. The MARTE Analysis Model is composed of three sub-packages. The Generic Quantitative Analysis Modeling (GQAM) package supports generic concepts for types of analysis based

on system execution behavior, which may be represented at different levels of detail.

GQAM is used to specify Schedulability Analysis Modeling (SAM) and Performance Analysis Modeling (PAM), offering facilities to annotate models with information required to perform schedulability or performance analysis.

3 Domain Characteristics

Traffic signals at road intersections are the major control measures applied in urban networks. The design of the control logic of a traffic signal must take care of efficiency and speed, but also of safety and security. The main purpose of traffic signals is to provide safe, efficient and fair crossing of the junction. When properly installed and operated, traffic signals provide a number of benefits [32], including the increase of the capacity of critical junction movements, reduction in the frequency and severity of accidents, and safe crossing of pedestrians and vehicles by providing interruptions in heavy streams. However, when poorly designed, traffic signals can cause excessive delays when cycle lengths are too long, increase the number of accidents (especially rear-end collisions), violations of red light, and lead to sub-optimal rerouting by drivers who want to avoid traffic signals.

Among the main advantages of traffic signals are the flexibility of the signaling scheme, the ability to provide priority treatment and the feasibility of coordinated control along streets. Modern traffic controllers implement signal timing and ensure that signal indications operate consistently and continuously in accordance with the pre-programmed phases and timing.

Many non-functional requirements are important for road traffic signals systems. These requirements do not have simple yes/no satisfaction criteria. Instead, it must be determined to what degree a non-functional requirement has been satisfied. Scalability is of great significance because there may occur changes in the road network, which may include new sensors and actuators to be accommodated into the system. Thus, new software objects may be inserted into the software architecture every time the network grows. This is important to allow the system to evolve over time.

Performance is also an issue, as there are many components to be controlled. Real-time constraints must be observed as data should be updated regularly. Example of available data are road traffic measurements information that are geographically distributed within the network. Availability of data is of considerable influence to system reliability and overall performance. Finally, flexibility is essential due to the possible change of component types, interfaces, and functionality.

4 Case Study

This case study is focused on the application of the MARTE profile together with UML in activities of modeling and design for a road traffic signal intersection control. The work was initially inspired by the modeling of an intersection

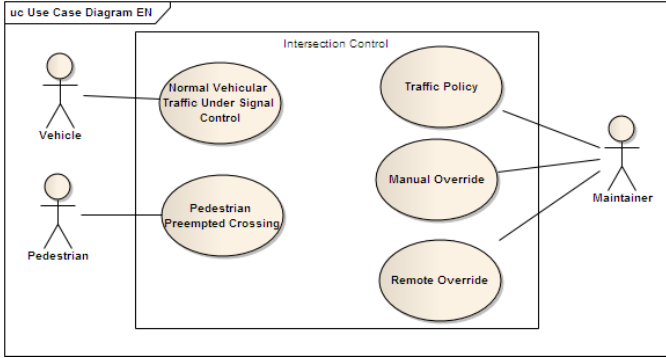


Fig. 2. Use Case Diagram

with UML proposed in [20]. The user requirements modeled in this case study are the requirements described in the following subsection. For a more realistic modeling of the system, the MARTE extensions that are best suitable to the main characteristics and problems described in Section 1 and 3 were applied. The following subsections describe the requirements, the structural design and the dynamic design for the road traffic signal intersection control.

4.1 User Requirements

It is important to notice that the user requirements presented in this subsection have high degree of abstraction. During system design, these requirements are refined into more detailed software requirements. The Use Case diagram corresponding to these requirements is illustrated in Figure 2. The following user requirements are used for system modeling.

1. The system shall control the traffic pattern of vehicles at the intersection.
2. The system shall control the traffic pattern of pedestrians at the intersection.
3. The system shall control the traffic flow of vehicles at all road sections related to the intersection.
4. The system shall control the traffic pattern of vehicles at all road sections related to the intersection.
5. The system shall allow fixed traffic control policy.
6. The system shall allow actuated traffic control policy.
7. The system shall allow adaptive traffic control policy.
8. The system shall allow green waves.
9. The system shall allow priorities for road sections.
10. The system shall detect the presence of pedestrians.
11. The system shall allow remote maintenance.
12. The system shall keep track of vehicle history at all road sections related to the intersection.
13. The system shall keep track of traffic policy during all year.
14. The system shall allow the implementation of new traffic policies.
15. The system shall keep track of accidents at the intersection.

4.2 Structural Design

Structural design is described in this article using the Class and the Deployment diagrams.

The class diagram supports the functional requirements and a set of key abstractions, taken in the form of objects. A UML Class diagram utilizing the extensions from MARTE is depicted in Figure 3. The structure of the class diagram is centralized mainly in the classes named *IntersectionController* and *Approach*. The intersection controller class is a singleton responsible for managing all features of the traffic signals control. It contains relationships with the approaches and phases. The approach class is responsible for managing the individual features of each approach in the intersection. It contains references to semaphores and push buttons.

The application of the MARTE profile in the class diagram is concentrated in new stereotypes. The stereotype `<<rtUnit>>` is used to represent active classes. The stereotype `<<storageResource>>` is used to represent entity classes. The stereotype `<<deviceResource>>` represents an external device that may be manipulated or invoked by the platform.

The stereotype `<<clockType>>` is related to time, which is a crucial feature of real time systems. The stereotype creates a new clock. In this case study two MARTE clock types are added in the class diagram. The first one is the predefined *IdealClock*. The *IdealClock* models the abstract and ideal time which is used in physical laws, i.e., it is a dense time. The *IdealClock* should be imported in models that refer to chronometric clock. The second is used to represent a scenario. The *Scenario* represents a scenario of utilization of the system, and the possible policies that can be applied in the traffic signals intersection control. The several values - references to quantities - presented in the class diagram are described using the VSL (Value Specification Language) notation [28].

The representation of resources and their further allocation are key features in the development of real-time systems. The definition of resources are important to deal with concurrency, distributed elements, and parallelism. The allocation of resources represents the allocation of functional application elements onto the available resources.

The most suitable UML diagram used to model resources and their allocation is the deployment diagram. The deployment diagram models the run-time configuration in a static view and visualizes the distribution of components in an application. Figure 4 depicts a deployment diagram to represent resources and Figure 5 a deployment diagram to represent allocation of resources.

Figure 4 is about the physical view of the software architecture of the intersection controller. It contains the main components of the system, such as the operating system, the application software, the network and also the physical components where the whole system is executed.

The representation of resources in MARTE is centralized in the stereotype `<<resource>>`. In this case study the `<<communicationMedia>>` is used in several parts of the deployment diagram. It represents the means to transport data from one location to another. Thus, its use represents the flow of data. The

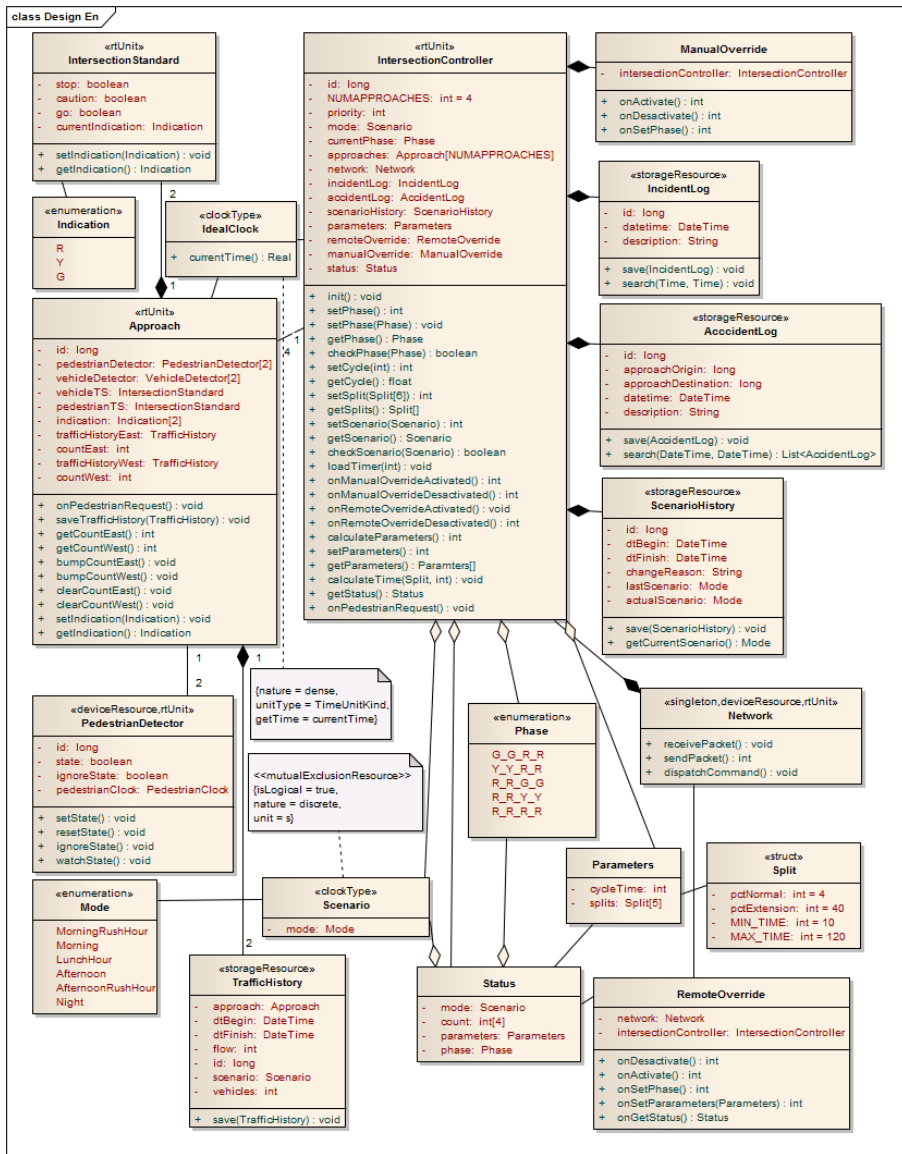


Fig. 3. Class Diagram with MARTE extensions

<<deviceResource>> and the <<storageResource>> stereotypes are the same as described in the class diagram. The <<computingResource>> represents either virtual or physical processing devices capable of storing and executing the software. In this case study, for example, the real-time operating system can be considered a <<computingResource>>.

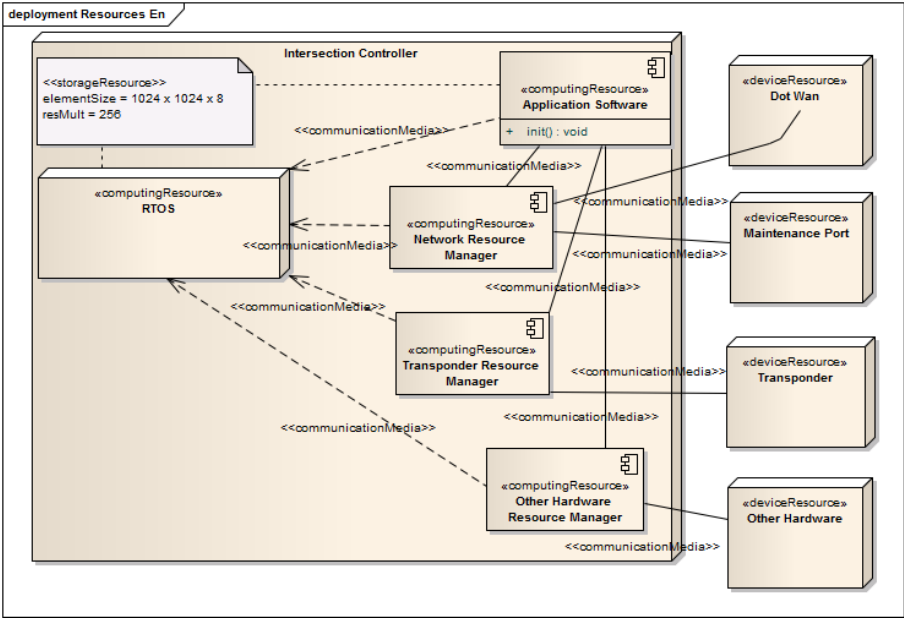


Fig. 4. Deployment Diagram with MARTE extensions for representation of resources

Figure 5 shows an example of allocation of resources with three layers. The first layer describes the application, the second layer represents a real-time operating system (RTOS) and the last layer shows the hardware parts.

The top layer is the application view. This view has the same components presented in Figure 4. The stereotype `<<allocated>>` is applied to named element that has at least one allocation relationship with another named element, and the tagged valued `{kind = application}` identifies an allocation end as being on the application side of the allocation.

The intermediate layer, the RTOS, is not the focus of this case study. It supports the allocations at different abstraction levels. The RTOS is related to the hardware through the stereotype `<<allocate>>` that is a bridge between elements from a logical context to elements in a more physical context. The nature is an enumeration type that defines literals used to specify the purpose of the allocation. The tagged value `{nature = timeScheduling}` indicates that the allocation consists of a temporal/behavioural ordering of the suppliers.

The lower layer in the diagram represents the hardware elements. The elements CPU, Memory, BUS and Disk are annotated with stereotypes described previously, such as `<<computingResource>>` and `<<communicationMedia>>`. All elements have the tagged value `{kind = executionPlatform}` that identifies an allocation end as being on the execution platform side of the allocation. Besides, this layer shows the relationship `<<allocate>>` between the components. This stereotype is attached with the tagged values `{nature = spatialDistribution}` and `{nature = timeScheduling}`. The *timeScheduling* was described previously

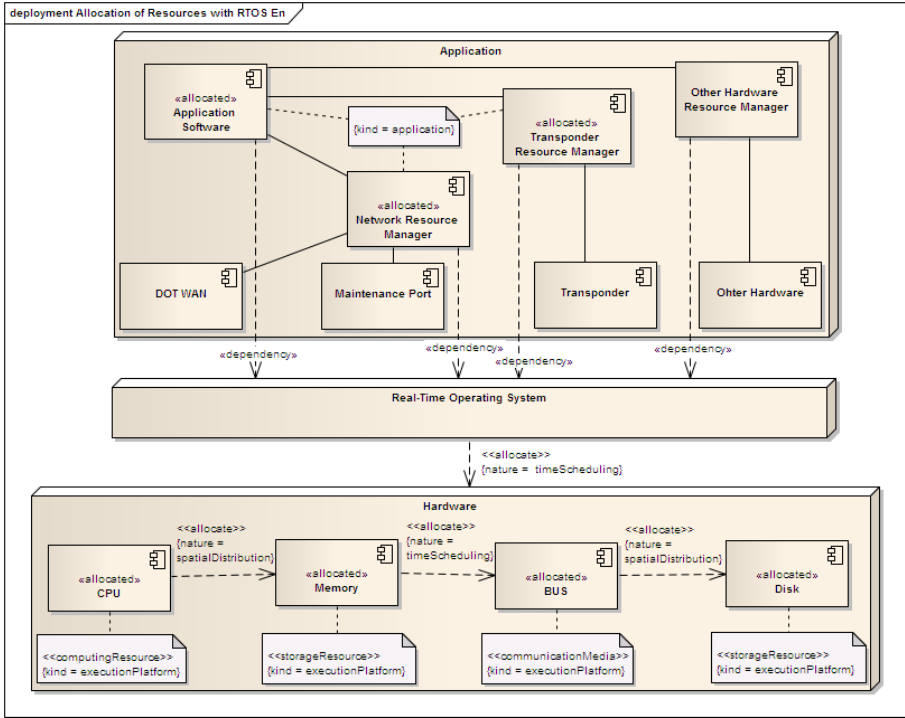


Fig. 5. Deployment Diagram with MARTE extensions for representation of allocation of resources

and the *spatialDistribution* indicates that the allocation consists of a tempo-ral/behavioural ordering of the suppliers.

4.3 Dynamic Design

Dynamic design is described in this article using the Sequence, State-Machine and Time diagrams.

The sequence diagram is applied to model the flow of logic within the system in a visual manner, which focuses on identifying the behavior within the system. A UML Sequence diagram utilizing the extensions from MARTE is depicted in Figure 6.

The sequence diagram shows the normal operation of the road traffic signal control. The diagram depicts the operation from the first action until the normal flow of the software.

For this purpose, the sequence diagram shows the *IntersectionController*, *Approach* and *IntersectionStandard* classes. The *IntersectionController* manages all system control. The *Approach* represents a specific approach in the intersection and the *IntersectionStandard* represents traffic signals. The vehicles’ traffic signals is represented by the object *vehicleTS* and the pedestrians’ traffic signals

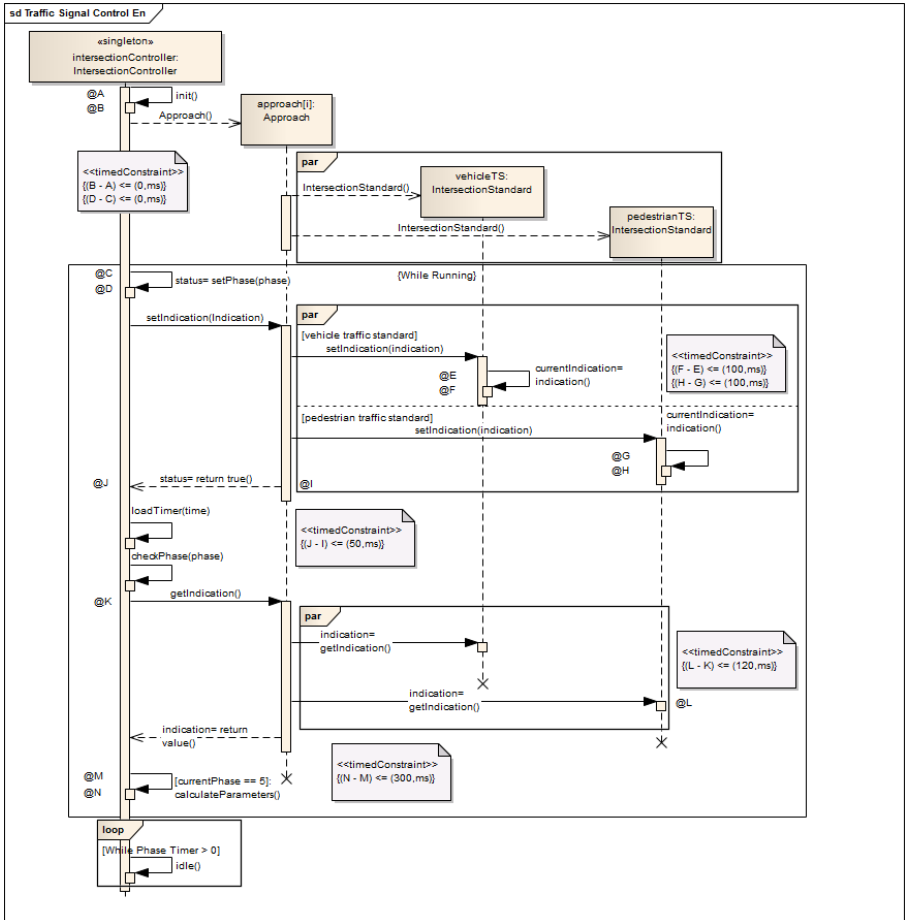


Fig. 6. Sequence Diagram with MARTE extensions

by the object *pedestrianTS*. The figure shows the initialization of the system, the creation of objects and the exchange of messages between the objects.

The sequence diagram focus is on time representation. In order to reach this purpose, the stereotype `<<timedConstraint>>` is used together with the VSL package. The *TimedConstraint* is an abstract superclass of *TimedInstantConstraint* and *TimedDurationConstraint*. It allows to constraint when an event may occur or constraint the duration of some execution or even constraint the temporal distance between occurrences of two events.

The state machine diagrams are useful in the real-time and embedded domain. They allow the description of a system behavior in terms of states and transitions between these states. A UML state machine diagram utilizing the extensions from MARTE is depicted in Figure 7. The state machine shows states *Waiting Signal* and *Pedestrian Crossing*. The first one represents a moment in which the

pedestrian is waiting for the signal. The last represents the moment in which the pedestrian is crossing the signal. The transitions between the two states represent the response to a pedestrian pressing the push button.

The application of the MARTE profile in the state machine diagram has new stereotypes as focus. The states are represented by the stereotype `<<mode>>`. It identifies an operational segment within the system execution. The transitions between states are represented by the stereotype `modeTransition`. It describes the modeled system under mode switching. The state machine is represented by two stereotypes. The `<<modeBehavior>>` specifies a set of mutually exclusive modes, i.e., only one mode can be active at a given instant. The stereotype `<<timedProcessing>>` represents activities that have known start and finish times or a known duration.

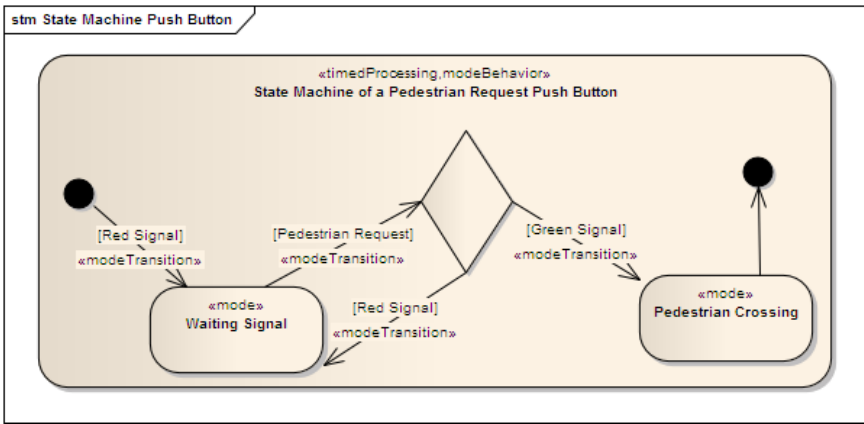


Fig. 7. State Machine with MARTE extensions

The time diagram is used to explore the behaviors of one or more objects throughout a given period of time. Timing diagrams are often used to design real-time systems. A UML timing diagram for traffic signals utilizing the extensions from MARTE is depicted in Figure 8. It shows that pedestrians have the right to cross when both traffic signals are at a red state.

5 Discussion

The comparison between UML, SPT and MARTE is not yet clear in present literature. In order to permit a broader view of the potential of MARTE, Table 1 presents a comparison between these OMG specifications based on specific criteria most common when designing real-time systems.

Table 1 shows the characteristics presented for each modeling language. The columns are annotated with the symbols \circ , \bullet e \bullet . Columns annotated with \circ means that the characteristic is not present in the modeling language or it is not satisfactory. Columns annotated with \bullet means that the characteristic is highly

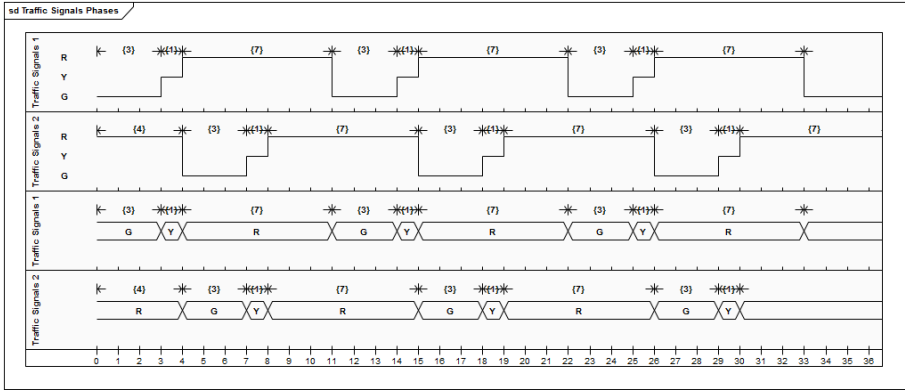


Fig. 8. Time Diagram with MARTE extensions

Table 1. Comparison of UML, SPT and MARTE

Criteria	UML [25]	SPT [23]	MARTE [28]
Modeling of time	○ [13] [10]	● [24]	●
Modeling of processes	○ [15]	●	● [10]
Modeling of resources	●	● [24]	● [10]
Modeling of resource allocation	● [15]	● [10] [24]	●
Modeling of hardware	○ [37] [26]	○ [10]	● [8]
Modeling of performance	○ [15]	● [30]	● [8]
Modeling of schedulability	○ [11] [15]	●	●
Modeling of run-time execution	○ [10]	● [24] [10]	● [10]
Modeling of task management	○ [6] [15]	●	● [10]
Modeling of requirements	○ [15]	○	○ [8]
CBSE	○	○ [30] [24]	● [8]
Alignment with UML 2.x	not applied	● [10] [24]	●
Formalism	○ [6] [15]	○	○ [8]
Consistency of diagrams	○ [17] [36] [15]	○	○ [18]
Ease of Use	● [15]	○ [24]	○
Software tools	●	○	○

present in the modeling language, i.e., its application is satisfactory. And columns annotated with ● means that the characteristics is not yet well-known, was not fully evaluated, or the language does not fully support the characteristic. Most evaluations were based on previous references, as indicated, and the languages specifications (UML [25], SPT [23], and MARTE [28]). Our own evaluation was considered only for the MARTE profile.

The main highlights to the table in this article are the references to UML and SPT problems and MARTE strengths and challenges. Overall, UML is weak for real-time systems design because the language is not capable of representing key real-time features, for instance, modeling time and processes. The table shows

the strengths of MARTE for modeling real-time systems. These strengths are described by previous works, as indicated in the table, and also by the application of MARTE in the case study shown in this article. For instance, design based on components (Component Based Software Engineering - CBSE) has improved with MARTE, as well as modeling time and resources.

Some challenges of MARTE are the same challenges of UML and SPT, such as increasing its formalism and improving consistency in diagrams. Another example is the modeling of requirements at an abstract level. The studied languages are tailored to model scenarios of requirements, often at a lower abstraction level. Another UML profile, the SysML, can fill this gap [35]. In addition, with too many stereotypes, constraints and tagged values, MARTE is a difficult language to master. Finally, there is a lack of software tools which can fully implement MARTE capabilities.

6 Conclusion

The design of real-time systems is a complex activity. Many modeling languages have been applied in this field. However, no single modeling language can be considered a standard in this area. The purpose of this article is to apply a new UML profile, MARTE, to model a distributed real-time system in the field of road traffic control, more specifically, traffic signals control. MARTE is used together with UML, complementing some UML aspects that are historically considered weak, including modeling of time, resources and processes. With MARTE, UML models are more specific. For instance, the modeling of resources has improved. However, MARTE also has weak characteristics. It is too complex, and it lacks software tools that implement the proposed stereotypes and extensions. Due to its complexity, mastering the MARTE language is a hard challenge.

It is worth to note that MARTE is an extensible profile, which means that further stereotypes can be created and added to the profile when necessary, which also increases its complexity. The industrial application of MARTE is still very incipient, and even in academia the language is still not well-known. This means that further evaluation and discussion on its applicability is necessary. A first attempt to compare UML, SPT and MARTE was shown in this article, but it still needs further evaluation in future research.

Acknowledgements. This work was supported by FAPEMIG (www.fapemig.br - EDITAL FAPEMIG 01/2011).

References

1. Addouche, N., Antoine, C., Montmain, J.: UML Models for Dependability Analysis of Real-Time Systems. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp. 5209–5214 (2004)
2. André, C., Mallet, F., de Simone, R.: Modeling Time(s). In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) MODELS 2007. LNCS, vol. 4735, pp. 559–573. Springer, Heidelberg (2007)

3. Bate, I., Hawkins, R., Toyn, I.: An Approach to Designing Safety Critical Systems using the Unified Modelling Language. In: Proceedings of the Workshop on Critical Systems Development with UML, pp. 3–17 (2003)
4. Belategi, L., Sagardui, G., Etxeberria, L.: MARTE Mechanisms to Model Variability When Analyzing Embedded Software Product Lines. In: Bosch, J., Lee, J. (eds.) SPLC 2010. LNCS, vol. 6287, pp. 466–470. Springer, Heidelberg (2010)
5. Bennett, A.J., Field, A.J.: Performance Engineering with the UML Profile for Schedulability, Performance and Time: A Case Study. In: Proceedings of the The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, pp. 67–75 (2004)
6. Berkenkotter, K.: Using UML 2.0 in Real-Time Development - A Critical Review. In: International Workshop on SVERTS: Specification and Validation of UML Models for Real Time and Embedded Systems (2003)
7. Berkenkötter, K., Bisanz, S., Hannemann, U., Peleska, J.: The HybridUML profile for UML 2.0. *International Journal on Software Tools for Technology* 8, 167–176 (2006)
8. Boutekkouk, F., Benmohammed, M., Bilavarn, S., Auguin, M.: UML 2.0 Profiles for Embedded Systems and Systems On a Chip (SOCs). *JOT (Journal of Object Technology)* 8(1), 135–157 (2009)
9. Cassandras, C.G., Lafortune, S.: Introduction to Discrete Event Systems. The International Series on Discrete Event Dynamic Systems. Kluwer Academic Publishers, Norwell (1999)
10. Demathieu, S., Thomas, F., André, C., Gérard, S., Terrier, F.: First Experiments Using the UML Profile for MARTE. In: Proceedings of the 2008 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing, pp. 50–57. IEEE Computer Society (2008)
11. Douglass, B.P.: Real Time UML: Advances in the UML for Real-Time Systems, 3rd edn. Addison Wesley Longman Publishing Co., Inc., Redwood City (2004)
12. Elhaji, M., Boulet, P., Tourki, R., Zitouni, A., Dekeyser, J.L., Meftali, S.: Modeling Networks-on-Chip at System Level with the MARTE UML profile. In: M-BED 2011, Grenoble, France (2011)
13. Graf, S., Ober, I., Ober, I.: A Real-Time Profile for UML. *International Journal on Software Tools for Technology Transfer* 8, 113–127 (2006)
14. Harel, D.: Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming* 8(3), 231–274 (1987)
15. Henderson-Sellers, B.: Uml - the good, the bad or the ugly? perspectives from a panel of experts. *Software and Systems Modeling* 4(1), 4–13 (2005)
16. Huang, Y.S., Liau, S.X., Jeng, M.D.: Modeling and Analysis of Traffic Light Controller using Statechart. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp. 557–562 (2010)
17. Jacobson, I.: Use cases - Yesterday, today, and tomorrow. *Software and System Modeling* 3(3), 210–220 (2004)
18. Jin, W., Wang, H., Zhu, M.: Modeling MARTE Sequence Diagram with Timing Pi-Calculus. In: ISORC, pp. 61–66 (2011)
19. Ranjini, K., Kanthimathi, A., Yasmine, Y.: Design of Adaptive Road Traffic Control System through Unified Modeling Language. *International Journal of Computer Applications* 14(7), 36–41 (2011)
20. Laplante, P.A.: Real-Time System Design and Analysis. John Wiley & Sons (2004)
21. List, G.F., Cetin, M.: Modeling Traffic Signal Control Using Petri Nets. *IEEE Transactions on Intelligent Transportation Systems* 5(3), 177–187 (2004)

22. Mraidha, C., Tanguy, Y., Jouvray, C., Terrier, F., Gerard, S.: An Execution Framework for MARTE-Based Models. In: 13th IEEE International Conference on Engineering of Complex Computer Systems, pp. 222–227 (2008)
23. OMG: UML Profile for Schedulability, Performance, and Time, Version 1.1. Tech. Rep. formal/2005-01-02, OMG (2005)
24. OMG: MARTE Tutorial: UML Profile for Develop for Real-Time and Embedded systems. Tech. Rep. formal/2007-03-28, OMG (2007)
25. OMG: OMG Unified Modeling Language (OMG UML) Superstructure, Version 2.3. Tech. Rep. formal/2010-05-03, OMG (2010)
26. OMG: Systems Modeling Language (SysML) - Version 1.2 (2010)
27. OMG: UML Profile for MARTE: Modeling and Analysis of Real-time Embedded Systems - version 1.1 (2010)
28. OMG: Uml profile for marte: Modeling and analysis of real-time embedded systems version, 1.1. Tech. Rep. formal/2011-06-02, OMG (2011)
29. OMG: Unified Modeling Language (UML): Superstructure - version 2.4.1 (2011)
30. Petriu, D.C., Woodside, M.: Extending the UML Profile for Schedulability Performance and Time (SPT) for Component-Based Systems (2004)
31. Quadri, I.R., Yu, H., Gamatié, A., Meftali, S., Dekeyser, J.L., Rutten, É.: Targeting Reconfigurable FPGA based SoCs using the MARTE UML profile: from high abstraction levels to code generation. *International Journal of Embedded Systems*, 18 (2010)
32. Roess, R.P., Prassas, E.S., McShane, W.R.: *Traffic Engineering*, 3rd edn. Prentice Hall, New Jersey (2003)
33. Soares, M.S.: Modeling and Analysis of Discrete Event Systems Using a Petri Net Component. In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 814–819 (2011)
34. Soares, M.S., Julia, S., Vrancken, J.: Real-time Scheduling of Batch Systems using Petri Nets and Linear Logic. *Journal of Systems and Software* 81(11), 1983–1996 (2008)
35. Soares, M.S., Vrancken, J.L.M., Verbraeck, A.: User Requirements Modeling and Analysis of Software-Intensive Systems. *Journal of Systems and Software* 84(2), 328–339 (2011)
36. Staines, A.S.: A Comparison of Software Analysis and Design Methods for Real Time Systems. *World Academy of Science, Engineering and Technology*, 55–59 (2005)
37. Süß, J., Fritzsion, P., Pop, A.: The Impreciseness of UML and Implications for ModelicaML. In: *Proceedings of the 2nd International Workshop on Equation-Based Object-Oriented Languages and Tools* (2008)
38. Thramboulidis, K.: Using UML in Control and Automation: A Model Driven Approach. In: *Proceedings of the IEEE International Conference on Industrial Informatics*, pp. 587–593 (2004)
39. Xu, J., Woodside, M., Petriu, D.: Performance Analysis of a Software Design Using the UML Profile for Schedulability, Performance, and Time. In: Kemper, P., Sanders, W.H. (eds.) *TOOLS 2003*. LNCS, vol. 2794, pp. 291–307. Springer, Heidelberg (2003)
40. Zeng, R., Li, G., Lin, L.: Adaptive Traffic Signals Control by Using Fuzzy Logic. In: *ICICIC 2007: Proceedings of the Second International Conference on Innovative Computing, Information and Control*, pp. 527–530 (2007)

Analysis of Techniques for Documenting User Requirements

Michel dos Santos Soares and Daniel Souza Cioquetta

Federal University of Uberlândia (UFU), Computing Faculty (FACOM),
Av. João Naves de Ávila, 2121, Bloco 1B
Uberlândia - Brazil
mics.soares@gmail.com, michel@facom.ufu.br

Abstract. A number of approaches were proposed in past years to document user requirements. Choosing the most suitable one is difficult, and frequently based on ad-hoc decision. In order to assist the way requirements engineers choose, an evaluation is necessary. The purpose of this paper is to analyze methods and languages used for user requirements documentation considering a number of criteria. This analysis was performed after extensive literature research and action research at companies that develop software-intensive systems. The objective is not to show how to find the best technique, the one that will perfectly suit all software projects. Instead, our purpose is to propose a critical view on a number of chosen techniques that might be useful for practitioners when choosing which technique to use on a specific project. The assumption is that stakeholders can benefit from knowing which techniques fit better a number of pre-determined evaluation criteria.

Keywords: User Requirements, Technique Evaluation, Requirements Documentation.

1 Introduction

Requirements for software are a collection of needs expressed by stakeholders respecting constraints under which the software must operate [44] [43]. Requirements can be classified in many ways. The classification used in this article is related to the level of detail. In this case, the two classes of requirements are user requirements and system requirements [50]. User requirements are high-level, abstract requirements based on end users' and other stakeholders' viewpoint. They are usually written using natural language, occasionally with the help of domain specific models such as mathematical equations, or even informal models not related to any method or language. The fundamental purpose of a user requirements specification is to document the needs and constraints gathered in order to later develop software based on those requirements.

The process by which requirements for systems and software products are gathered, analyzed, documented and managed throughout the development life cycle is called Requirements Engineering (RE) [50]. RE is a very influential phase

in the software life cycle. According to the SWEBOK [1], it concerns Software Design, Software Testing, Software Maintenance, Software Configuration Management, Software Engineering Management, Software Engineering Process, and Software Quality Knowledge Areas. RE is considered by many authors the most critical phase within the development of software [33] [35] [18] [39]. Already in 1973, Boehm suggested that errors in requirements could be up to 100 times more expensive to fix than errors introduced during implementation [10]. According to Brooks [13], knowing what to build, which includes requirements elicitation and further technical specification, is the most difficult phase in software development. Dealing with ever-changing requirements is considered the real problem of Software Engineering [9]. Studies conducted by the Standish Group [54] and other researchers and practitioners [25] [27] found that the main factors for problems with software projects (cost overruns, delays, user dissatisfaction) are related to requirements issues, such as lack of user input, incomplete requirements specifications, uncontrolled requirements changing, and unclear objectives.

RE can be divided into two main groups of activities [42]: i) requirements development, including activities such as eliciting, documenting, analyzing, and validating requirements, and ii) requirements management, including activities related to maintenance, such as tracing and change management of requirements. This article is about user requirements development, mainly the activities of documenting and analyzing user requirements for developing software.

This article can be compared to others which main theme is how to select RE techniques. For instance, 46 techniques are investigated to create a methodology to select RE techniques in [32]. The authors considered a very broad RE context, comparing techniques at different levels of detail, and with very different purposes. For instance, they compared Petri nets, interviews, and tests. In another study, Maiden and Rugg [38] proposed a framework which provides general help for requirements engineers to select methods for requirements acquisition. In a well-known RE book, Sommerville and Kotonya [51] proposed high level, general attributes which can be used for the evaluation and selection of RE techniques in general. In [14], Browne and Ramesh proposed an idea of technique selection for requirements elicitation. This technique selection method is built on the human cognitive model.

There are two important differences between our paper and others previously cited. The first one is that the focus of our paper is only on requirements documentation at the user level. Therefore, we are not concerned with other requirements levels (implementation, domain) and we do not compare techniques used at different levels. The second one is that we did our research considering theoretical aspects but also practical ones. For that reason, we have also used Action Research as a research instrument [4] [6]. The reason is because we wanted to try out our theories with practitioners in real situations and real organizations. We performed interviews and practical application through Action Research in two companies. During the interviews, we asked questions about the software development methodology and the technical environment.

The reminder of the paper is organized as follows. Section 2 presents a (non-exhaustive) list of eight methods and techniques applied on the activity of requirements documentation. Section 3 brings a list of criteria used for evaluating the methods presented in the previous section. This list of criteria is based on interviews with practitioners and researchers and on literature review. The resulting evaluation table is presented in Section 4, and conclusion and on going work in Section 5.

2 Review on Techniques for User Requirements Documentation

There are a number of languages and methods used in all activities of RE [44] [43] [50]. In this article we are interested only on requirements documentation. We start our investigation from the list presented in [32].

From this list, we selected the techniques used for documenting and specifying requirements, with exception to formal methods. The reason for this decision is that the applicability and suitability of formal methods in practice is still highly debated in the Software Engineering community [12] [36] [2] [57]. Practitioners often consider formal methods inadequate, restricted to critical systems, too expensive, insufficient, and too difficult [15] [21]. Real-world examples are still lacking or the application scope is generally considered limited [56]. However, the most important factor to exclude formal methods in this research is because formal methods are more likely to be applied to the design and verification phases of software development, not specifications of user requirements. A number of other techniques were not taken into account in this study because we would like to evaluate the most common techniques, the ones that are regularly used in practice or at least are well-known in academia. The selected techniques are presented in the following paragraphs. For each item of the list, we propose a number of evaluation criteria (Section 3).

The most common approach to document user requirements is to write them using *Natural language* [37]. The advantages are that natural language is simple and is the main mean of communication between stakeholders. However, problems such as imprecision, misunderstandings, ambiguity and inconsistency are common when natural language is used [34]. As a matter of fact, these problems occur frequently when natural language is the only means of description of requirements.

Structured Natural Languages are used with the purpose of giving more structure to requirements documents. Nevertheless, structured natural languages are neither formal nor graphical, and can be too much oriented to algorithms and specific programming languages [17]. Other collateral effects are that structured specifications may limit too early the programmers' freedom to coding.

Viewpoints are an approach for requirements elicitation in which requirements are organized into a number of views, giving structure to the processes of eliciting and specifying software requirements. As it is almost impossible to recognize all information about requirements considering only one perspective,

it is necessary to collect and to organize requirements at a number of different viewpoints. The technique is largely used in industry [53]. Viewpoint analysis is considered a simple and flexible technique which supports the grouping of requirements, requirements management, verification of inconsistencies, and requirements traceability [52] [24]. A key strength of viewpoint-based documentation is that it recognizes multiple perspectives and provides a framework for discovering conflicts in requirements proposed by different stakeholders [50]. Another characteristic of viewpoints is that it allows the management of inconsistencies by providing support to detect and to solve them [23]. Typically, each viewpoint provides different types of requirements. One issue with viewpoints is that, in practice, for a high number of identified viewpoints, it may be difficult to prioritize requirements [50].

Decision tables [28] provide a notation that translates actions and conditions into a tabular format. The table can be used as a machine-readable input to a table-driven algorithm. This technique is useful when a complex set of conditions and actions are encountered within a component [43].

A well-known diagram used for requirements modeling is the *Use Case diagram*. Even before UML emerged as the main Software Engineering modeling language, Use Cases were already a common practice for graphically representing functional requirements in methodologies such as Object-Oriented Software Engineering (OOSE) [30]. Their popularity can be explained due to their simplicity, making them act as a bridge between technical and business stakeholders. The compact graphical nature is useful to represent requirements that may be expanded to several pages. Use Cases also have some disadvantages and problems [47]. They are applied mainly to model functional requirements and are not very helpful for other types of requirements, such as non-functional ones. Use Case diagrams lack well-defined semantics, which may lead to differences in interpretation by stakeholders. For instance, the *include* and the *extend* relationships are considered similar, or even the inverse of each other [31]. In addition, Use Cases may be misused, when too much detail is added, which may incorrectly transform the diagrams into flowcharts or make them difficult to comprehend.

User Stories have been used as part of the eXtreme Programming (XP) [7] agile methodology. They can be written by the customer using non-technical terminology in the format of sentences using natural language. Although XP offers some advantages in the RE process in general, such as user involvement and defined formats for user requirements and tasks, requirements are still loosely related, not graphically specified, and oriented to a specific methodology.

Two SysML diagrams are distinguished as useful mainly for RE activities: the *SysML Requirements* diagram and the *SysML Use Case* diagram [41]. One interesting feature of the SysML Requirements diagram is the flexibility to model other types of requirements besides the functional ones, such as non-functional requirements. The SysML Use Case diagram is derived from the UML Use Case diagram without important modifications. In addition to these diagrams, *SysML Tables* can be used to represent requirements in a tabular format. Tabular representations are often used in SysML but are not considered part of the diagram

taxonomy [41]. Important decisions on requirements and the correspondent models are better justified when traceability is given proper attention. SysML Tables allows the representation of requirements, their properties and relationships in a tabular format. One way to manage the requirements traceability in SysML is by using requirements tables.

The SysML Requirements diagram helps in organizing requirements and also shows explicitly the various kinds of relationships between different requirements. The diagram is useful to standardize the way to specify requirements through a defined semantics. As a direct consequence, SysML allows the representation of requirements as model elements, which means that requirements are part of the system architecture [5]. A SysML Requirement can also appear on other diagrams to show its relationship to design. With the SysML Requirements diagram, visualization techniques are applied from the early phases of system development.

The focus of goal-oriented modeling shifts from what and how (data and processes) as addressed by traditional analysis to who and why (the actors and the goals they wish to achieve). Goal-oriented modeling addresses the early analysis or requirements elicitation. *i** is one of the most widely used goal modeling languages. Its graphical notation is considered clear and easy to use and understand [58]. The strengths of *i** are the simple semantics and the graphical notation. The notation supports the modeling of dependencies between actors [58]. These dependencies can be classified into diverse types. However, the adoption of *i** in industry is still very low, even after many years of the introduction of this technique [19]. *i** lacks explicit design rationale for its graphical conventions. Its semantic constructs and grammatical rules are defined using natural language [40], which leads to problems of inconsistency, ambiguity, and incompleteness.

3 Criteria Used for Evaluation

The criteria considered for evaluating the techniques and languages used for documenting requirements are given in this section. These criteria are based on interviews with practitioners and researchers [46] [49], on Action Research performed at companies that develop software [48], on the IEEE Recommended Practice for Software Requirements Specifications [29] criteria for a good requirements document, and on classical textbooks on Software Engineering and Requirements Engineering [44] [43] [50].

The interviews and the practical application through Action Research were performed at two companies with diverse characteristics. The first one is a large financial company that develops software to a wide variety of technical environments, such as web and mainframe software applications. The company not only develops software internally, but also hire consulting companies to develop parts or entire software systems. The second company is a small consulting company that develops software systems in the field of road traffic management.

Interviews were performed with project managers and senior developers. During the interviews, we asked general questions about the software development

methodology and technical environment. Specifically about user requirements, we asked questions such as “What criteria do you use to evaluate a user requirements technique”, “What would you expect from techniques to document user requirements”, and “What techniques do you use to document user requirements in your company”. The chosen criteria are presented in the following paragraphs.

Graphical modeling Graphical models are an advantage over strictly text-based approaches in terms of facility to communicating and to understanding requirements. Naturally, stakeholders must have knowledge about the graphical notation, otherwise the graphical models would not make sense, or even worse, would be considered to have different meaning by stakeholders.

Human readable As our approach is to evaluate techniques to document requirements at the user level, the more human readable models the technique can provide, the better.

Independent towards methodology User requirements are independent of a specific technology and are considered to have high abstraction level. At this level, stakeholders are concerned with functionalities and properties, and not so much about how to design and implement software. Therefore, independency towards design methodologies allows good separation of concerns and provide freedom during design and implementation.

Relationship requirements-requirements It is well-known by software engineers that requirements are related to each other. The survey [45] introduces the discipline of Requirements Interaction Management (RIM), which is concerned with analysis and management of dependencies among requirements. These interactions affect various software development activities, such as release planning, change management and reuse. Thus, it is almost impossible to plan systems releases based only on the highest priority requirements, without considering which requirements are related to each other.

In addition to identifying the relationship between requirements, it is also interesting to identify the *type of relationship between requirements*. For instance, one can be interested in knowing how a requirement is affected when a related requirement is changed or even deleted.

Identify and represent types of requirements Most commonly modeling languages are applicable only to model functional requirements. Despite their importance, non-functional requirements are usually not properly addressed in requirements modeling languages. This is also true for other types of requirements, such as external and domain ones.

Priority between requirements From a project management point of view, one important characteristic of a requirement is its priority. Prioritizing requirements is an important activity in RE [20]. The purpose is to provide an indication of the order in which requirements should be addressed. The reason is that implementing all requirements at once might be unattractive. It may take too long, and stakeholders normally want to use the new software as soon as possible. Therefore, prioritizing requirements is a possible solution to plan software releases.

Grouping related requirements As a consequence of the relationship between requirements, semantically related requirements can be grouped. Hence, these groups can be considered high cohesion units. The advantage is that grouping requirements help in giving shape to the software architecture already in the early phases of development, which is in accordance with the concept of architecture first [11].

Flexible With this criteria, we want to know how flexible a technique is to document user requirements. Probably a balance is the best option here. A technique with high degree of flexibility will allow designers to create requirements documents with poor structure. On the other hand, a very strict technique will inhibit designers at an early phase of software development.

Ranking requirements by stability A great source for problems in software engineering is requirements changing. Therefore, knowing how stable a requirement is, i.e., how ready it is for further design phases, is essential.

Solve ambiguity Ambiguity in requirements is a major cause for misunderstandings between stakeholders and designers. Thus, modeling languages should provide well-defined semantics, which might increase *machine readability* with the drawback of diminishing human readability.

Well-defined semantics Having a technique that is based on well-defined semantics helps in the production of software requirements specifications with less ambiguity.

Verifiable According to [29], one characteristic of a good software requirements specification is that it should be verifiable. A requirement is verifiable if there exists some finite cost-effective process with which a person or machine can check that the software product meets the requirement. In order to facilitate the writing of verifiable requirements, the technique should allow the use of concrete terms and measurable quantities.

Expressibility Expressibility is the ability to represent requirements. The technique should provide means to express requirements in such a way that stakeholders from diverse backgrounds can understand, and such that communication of requirements to stakeholders is clear.

Ability in requirements management Stakeholders frequently change requirements due to various factors. For example, stakeholders may be unsure about their own needs in the beginning of a project, and laws and business processes may change. The main issue is not related to changes in requirements, but to uncontrolled changes. Through correct requirements management, whenever stakeholders ask for changes in requirements, developers have the possibility to uncover where and how this change will impact the system design. Hence, the technique should be capable of facilitating changes in requirements.

Simplicity It is important to note that, at least in theory, simplicity comes together with high potential of use (see Technology Acceptance Model [55]). However, it is hard to evaluate simplicity as it is a quite subjective quality. One example is the Use Case diagram, which is considered simple enough to be understandable by most stakeholders [30] [31], but also only understandable by

stakeholders with technical background [47]. During our interviews, most of the time the quality of simplicity was very debated.

Ability to facilitate communication Documenting requirements for future reference and to communicate requirements with stakeholders are core objectives of requirements documentation. Therefore, the technique should facilitate communication between stakeholders.

Technique maturity The gap between the development of new academic methods, techniques and processes and their actual application in industry is common in Systems and Software Engineering [16]. The challenge is not only to develop better theories, but also to effectively introduce and use these theories in practice. Thus, one concern during our research was on identifying how mature the technique is.

Traceability A requirement is traceable if its origin is clear and if it facilitates its referencing in future development or enhancement documentation. Traceability of a requirement is especially important when the software product enters the operation and maintenance phase. As code and design documents are modified, it is essential to be able to ascertain the complete set of requirements that may be affected by those modifications. Therefore, the technique should be able to enable *traceability between requirements*, and *traceability between requirements and design*.

Used in industry A great challenge with techniques, languages, and methods in software engineering in general, and in RE in particular, is to put them to work in practice. Often a successful technique from the industry point-of-view is poorly defined or not appreciated in academia. For instance, UML has many well-known flaws. The language is considered too informal and ambiguous [8] [26]. There are too many diagrams, with some rarely used in practice [22], making it more difficult to choose which one should be used in a specific situation [3]. Despite all of these issues, UML is currently the most used modeling language in industry. We are interested in knowing how well-established in industry the technique is.

4 Resulting Table

Table 1 is a result of the evaluation of the proposed techniques against the defined criteria. In the table, NL stands for Natural Language, SNL stands for Structured Natural Language, VP stands for Viewpoint-based Documentation, DT stands for Decision Table, UC stands for Use Cases (both SysML and UML), SR stands for SysML Requirements diagram, ST stands for SysML Tables, and i* stands for the goal-oriented modeling i*.

We classified the entries as fully supported (●), half supported (◐), or not supported (○) (or not easily supported, or poorly supported). Some entries were difficult to evaluate and generated debates. Therefore, we decided to use the symbol “?” when in doubt, i.e., literature research and the interviews about the technique for that specific criteria were not conclusive.

Based on the characteristics of each technique, and on the needs expressed by developers and managers, they can evaluate and select a suitable technique.

Supposing that “Graphical modeling” and “Relationship between requirements” are the most relevant criteria, then the stakeholders might chose between the SysML Requirements diagram and i* as most appropriate techniques in this case, instead of structured natural languages, for instance. Therefore, the table can be seen as a guideline to help stakeholders to select which technique suits better their purpose. A decision support tool based on this research was developed in order to help stakeholders in taking a decision based on the characteristics of the project at hand.

It should be clear that techniques are not mutually exclusive. Two or more techniques may be used in combination. That depends on stakeholders’ interest, on the proper characteristics of each technique, and on characteristics of the software to be developed. In addition, when two techniques are considered equally supporting a set of chosen criteria, the decision should be taken based on other principle. For instance, previous knowledge of the technique by the developers should be taken into account.

Table 1. Table relating techniques and evaluation criteria

List of requirements	NL	SNL	VP	DT	UC	SR	ST	i*
Graphical Modeling	○	○	●	◐	●	●	◐	●
Human readable	●	◐	●	●	◐	◐	●	◐
Independent towards methodology	●	◐	●	●	◐	●	●	○
Relationship requirements-requirements	○	○	◐	○	◐	●	●	●
Type of relationship between requirements	○	○	?	○	◐	●	●	○
Represent types of requirements	●	●	○	?	○	●	●	◐
Priority between requirements	●	●	◐	○	○	●	●	○
Grouping related requirements	●	●	●	●	●	●	●	●
Flexible	●	○	●	●	◐	◐	●	●
Ranking requirements by stability	○	○	?	○	○	●	◐	○
Solve ambiguity	○	◐	◐	●	○	◐	●	○
Well-defined semantics	○	◐	○	◐	○	◐	●	○
Verifiable	○	◐	○	●	◐	●	●	◐
Expressibility	●	◐	●	●	◐	●	◐	●
Ability in requirements management	○	○	●	●	◐	●	●	●
Simplicity	●	◐	●	●	◐	?	●	◐
Ability to facilitate communication	◐	◐	●	●	◐	◐	●	◐
Technique maturity	●	●	●	●	●	◐	◐	●
Traceability: Requirements-Requirements	○	○	○	◐	◐	●	●	●
Traceability: Requirements-Design	○	◐	?	●	●	●	●	?
Used in industry	●	●	●	●	●	◐	◐	◐

5 Conclusion

Chosing one technique for Requirements Engineering among the great variety proposed in past years is a challenge that developers have to face. Currently, the

decision is frequently based on ad-hoc criteria. The main purpose of this article is to assist requirements engineers on taking such as important decision.

In this article the evaluation of a number of techniques used for documenting user requirements against a number of criteria is proposed. The source for evaluation was literature review, action research on two companies, and interviews with practitioners, which help to determine the criteria for the evaluation. The purpose was not to come to conclusions on which technique is better. Instead, this research can be seen as a guideline to chose techniques for documenting user requirements. Therefore, based on the project at hand and on criteria given by the proper stakeholders, some techniques may be considered more suitable than others. Stakeholders can benefit from knowing which techniques fit better a number of pre-determined evaluation criteria. This research shed a light on the capabilities of the most common techniques applied to model user requirements.

Ongoing research concerns the evaluation of the decision support tool developed in order to help stakeholders to choose a technique for Requirements Engineering documentation based on project characteristics and the stakeholders interests.

Acknowledgements. This work was supported by PROPP-UFU (<http://www.propp.ufu.br/site/>).

References

1. Abran, A., Bourque, P., Dupuis, R., Moore, J.W., Tripp, L.L. (eds.): Guide to the Software Engineering Body of Knowledge - SWEBOOK, 2004 version edn. IEEE Press, Piscataway (2004)
2. Abrial, J.R.: Formal Methods: Theory Becoming Practice. *Journal of Universal Computer Science* 13(5), 619–628 (2007)
3. Anda, B., Hansen, K., Gullesten, I., Thorsen, H.K.: Experiences from Introducing UML-based Development in a Large Safety-Critical Project. *Empirical Software Engineering* 11(4), 555–581 (2006)
4. Avison, D.E., Lau, F., Myers, M.D., Nielsen, P.A.: Action Research. *Communications of the ACM* 42(1), 94–97 (1999)
5. Balmelli, L., Brown, D., Cantor, M., Mott, M.: Model-driven Systems Development. *IBM Systems Journal* 45(3), 569–586 (2006)
6. Baskerville, R.: Investigating Information Systems with Action Research. *Communications of the AIS* 2(4), 1–32 (1999)
7. Beck, K.: *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, Boston (1999)
8. Beneken, G., Hammerschall, U., Broy, M., Cengarle, M.V., Jürjens, J., Rumpe, B., Schoenmakers, M.: Componentware - State of the Art 2003. In: *Proceedings of the CUE Workshop Venedig* (2003)
9. Berry, D.M.: The Inevitable Pain of Software Development: Why There Is No Silver Bullet. In: Wirsing, M., Knapp, A., Balsamo, S. (eds.) *RISSEF 2002*. LNCS, vol. 2941, pp. 50–74. Springer, Heidelberg (2004)
10. Boehm, B.W.: Software and Its Impact: A Quantitative Assessment. *Datamation* 19(5), 48–59 (1973)

11. Booch, G.: The Economics of Architecture-First. *IEEE Software* 24, 18–20 (2007)
12. Bowen, J.P., Hinchey, M.G.: Seven More Myths of Formal Methods. *IEEE Software* 12(4), 34–41 (1995)
13. Brooks, F.P.: No Silver Bullet: Essence and Accidents of Software Engineering. *Computer* 20(4), 10–19 (1987)
14. Browne, G.J., Ramesh, V.: Improving Information Requirements Determination: A Cognitive Perspective. *Information Management* 39(8), 625–645 (2002)
15. Broy, M.: From “Formal Methods” to System Modeling. In: Jones, C.B., Liu, Z., Woodcock, J. (eds.) *Formal Methods and Hybrid Real-Time Systems*. LNCS, vol. 4700, pp. 24–44. Springer, Heidelberg (2007)
16. Connor, A.M., Buchan, J., Petrova, K.: Bridging the Research-Practice Gap in Requirements Engineering through Effective Teaching and Peer Learning. In: *ITNG 2009: Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations*, pp. 678–683. IEEE Computer Society (2009)
17. Cooper, K., Ito, M.: Formalizing a Structured Natural Language Requirements Specification Notation. In: *Proceedings of the International Council on Systems Engineering Symposium*, vol. CDROM index 1.6.2, Las Vegas, Nevada, USA, pp. 1–8 (2002)
18. Damian, D., Zowghi, D., Vaidyanathasamy, L., Pal, Y.: An Industrial Case Study of Immediate Benefits of Requirements Engineering Process Improvement at the Australian Center for Unisys Software. *Empirical Software Engineering* 9(1-2), 45–75 (2004)
19. Davies, I., Green, P., Rosemann, M., Indulska, M., Gallo, S.: How Do Practitioners Use Conceptual Modelling in Practice? *Data Knowledge Engineering* 58(3), 358–380 (2006)
20. Davis, A.M.: The Art of Requirements Triage. *Computer* 36(3), 42–49 (2003)
21. Davis, J.F.: The Affordable Application of Formal Methods to Software Engineering. *ACM SIGAda Ada Letters* XXV(4), 57–62 (2005)
22. Dobing, B., Parsons, J.: How UML is Used. *Communications of the ACM* 49(5), 109–113 (2006)
23. Easterbrook, S., Nuseibeh, B.: Using ViewPoints for inconsistency management. *Software Engineering Journal* 11(1), 31–43 (1996)
24. Easterbrook, S., Nuseibeh, B.: Using ViewPoints for inconsistency management. *Software Engineering Journal* 11(1), 31–43 (1996)
25. van Genuchten, M.: Why is Software Late? An Empirical Study of Reasons For Delay in Software Development. *IEEE Transactions on Software Engineering* 17(6), 582–590 (1991)
26. Henderson-Sellers, B.: UML - the Good, the Bad or the Ugly? Perspectives from a panel of experts. *Software and System Modeling* 4(1), 4–13 (2005)
27. Hofmann, H.F., Lehner, F.: Requirements Engineering as a Success Factor in Software Projects. *IEEE Software* 18(4), 58–66 (2001)
28. Hurley, R.B.: *Decision Tables in Software Engineering*. John Wiley & Sons, Inc., New York (1983)
29. IEEE: *IEEE Recommended Practice for Software Requirements Specifications*. Tech. rep. (1998)
30. Jacobson, I.: *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley Professional, Reading (1992)
31. Jacobson, I.: Use Cases - Yesterday, Today, and Tomorrow. *Software and System Modeling* 3(3), 210–220 (2004)

32. Jiang, L., Eberlein, A., Far, B., Mousavi, M.: A Methodology for the Selection of Requirements Engineering Techniques. *Software and Systems Modeling* 7, 303–328 (2007)
33. Juristo, N., Moreno, A.M., Silva, A.: Is the European Industry Moving Toward Solving Requirements Engineering Problems? *IEEE Software* 19(6), 70–77 (2002)
34. Kamsties, E.: Understanding Ambiguity in Requirements Engineering. In: Aurum, A., Wohlin, C. (eds.) *Engineering and Managing Software Requirements*, pp. 245–266. Springer, Berlin (2005)
35. Komi-Sirviö, S., Tihinen, M.: Great Challenges and Opportunities of Distributed Software Development - An Industrial Survey. In: *Proceedings of the Fifteenth International Conference on Software Engineering and Knowledge Engineering (SEKE 2003)*, pp. 489–496 (2003)
36. Larsen, P.G., Fitzgerald, J., Brookes, T.: Applying Formal Specification in Industry. *IEEE Software* 13(3), 48–56 (1996)
37. Luisa, M., Mariangela, F., Pierluigi, I.: Market Research for Requirements Analysis Using Linguistic Tools. *Requirements Engineering* 9(1), 40–56 (2004)
38. Maiden, N., Rugg, G.: ACRE: Selecting Methods for Requirements Acquisition. *Software Engineering Journal* 11(3), 183–192 (1996)
39. Minor, O., Armarego, J.: Requirements Engineering: a Close Look at Industry Needs and Model Curricula. *Australian Journal of Information Systems* 13(1), 192–208 (2005)
40. Moody, D.L., Heymans, P., Raimundas Matulevičius, R.: Visual Syntax Does Matter: Improving the Cognitive Effectiveness of the i* Visual Notation. *Requirements Engineering* 15(2), 141–175 (2010)
41. OMG: *Systems Modeling Language (SysML) - Version 1.2* (2010)
42. Parviainen, P., Tihinen, M., Lormans, M., van Solingen, R.: *Requirements Engineering: Dealing with the Complexity of Sociotechnical Systems Development.*, ch. 1, pp. 1–20. IdeaGroup Inc. (2004)
43. Pressman, R.S.: *Software Engineering: A Practitioner's Approach*, 7th edn. McGraw-Hill, Inc., New York (2010)
44. Robertson, S., Robertson, J.: *Mastering the Requirements Process*, 2nd edn. Addison-Wesley Professional, ACM Press/Addison-Wesley Publishing Co., New York (2006)
45. Robinson, W.N., Pawlowski, S.D., Volkov, V.: Requirements Interaction Management. *ACM Computing Surveys* 35(2), 132–190 (2003)
46. Sharon, I., Soares, M.S., Barjis, J., van den Berg, J., Vrancken, J.L.M.: A Decision Framework for Selecting a Suitable Software Development Process. In: *Proceedings of ICEIS 2010, 12th International Conference on Enterprise Information Systems*. vol. 3, pp. 34–43 (2010)
47. Simons, A.J.H.: Use Cases Considered Harmful. In: *TOOLS 1999: Proceedings of the Technology of Object-Oriented Languages and Systems*, pp. 194–203 (1999)
48. Soares, M.S., Vrancken, J.L.M.: Evaluation of UML in Practice - Experiences in a Traffic Management Systems Company. In: *Proceedings of 12th International Conference on Enterprise Information Systems, ICEIS 2010*, pp. 313–319 (2009)
49. Soares, M.S., Vrancken, J.L.M., Verbraeck, A.: User Requirements Modeling and Analysis of Software-Intensive Systems. *Journal of Systems and Software* 84(2), 328–339 (2011)
50. Sommerville, I.: *Software Engineering*, 9th edn. Addison-Wesley, Essex (2010)
51. Sommerville, I., Kotonya, G.: *Requirements Engineering: Processes and Techniques*. John Wiley & Sons, Inc., New York (1998)

52. Sommerville, I., Sawyer, P., Viller, S.: Managing process inconsistency using viewpoints. *IEEE Transactions on Software Engineering* 25, 784–799 (1999)
53. Sommerville, I., Sawyer, P., Viller, S.: Viewpoints for Requirements Elicitation: A Practical Approach. In: *Proceedings of the 3rd International Conference on Requirements Engineering: Putting Requirements Engineering to Practice, ICRE 1998*, pp. 74–81 (1998)
54. The Standish Group: *CHAOS Chronicles v3.0*. Tech. rep., The Standish Group (2003) (last accessed on the August 20, 2009)
55. Venkatesh, V., Bala, H.: Technology Acceptance Model 3 and a Research Agenda on Interventions. *Decision Sciences* 39(2), 273–315 (2008)
56. Wassyng, A., Lawford, M.: Lessons Learned from a Successful Implementation of Formal Methods in an Industrial Project. In: Araki, K., Gnesi, S., Mandrioli, D. (eds.) *FME 2003*. LNCS, vol. 2805, pp. 133–153. Springer, Heidelberg (2003)
57. Woodcock, J., Larsen, P.G., Bicarregui, J., Fitzgerald, J.: *Formal Methods: Practice and Experience*. *ACM Computing Surveys* 41(4), 1–36 (2009)
58. Yu, E.S.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In: *International Symposium on Requirements Engineering*, pp. 226–235 (1997)

Predicting Web Service Maintainability via Object-Oriented Metrics: A Statistics-Based Approach

José Luis Ordiales Coscia², Marco Crasso^{1,2,3}, Cristian Mateos^{1,2,3},
Alejandro Zunino^{1,2,3}, and Sanjay Misra⁴

¹ ISISTAN Research Institute

² UNICEN University

³ Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)

⁴ Department of Computer Engineering, Atilim University, Ankara, Turkey

Abstract. The Service-Oriented Computing paradigm enables the construction of distributed systems by assembling loosely coupled pieces of software called services, which have clear interfaces to their functionalities. Service interface descriptions have many aspects, such as complexity and quality, all of which can be measured. This paper presents empirical evidence showing that services interfaces maintainability can be predicted by applying traditional software metrics in service implementations. A total of 11 source code level metrics and 5 service interface metrics have been statistically correlated using 154 real world services.

Keywords: Service-Oriented Computing, Web Services, Code-First, Web Service Maintainability, Object-Oriented Metrics, Web Service Maintainability Prediction.

1 Introduction

Service-Oriented Computing (SOC) is a paradigm that allows engineers to build new software by composing loosely coupled pieces of existing software called *services*. A distinguishing feature of SOC is that services may be provided by third-parties who only expose services interfaces to the outer world. By means of these interfaces, potential consumers can determine what a service does from a functional perspective and remotely invoke it from their new applications.

The advances in distributed system technologies have caused engineers to materialize SOC in environments with higher levels of distribution and heterogeneity. The availability of broadband and ubiquitous connections enable to reach the Internet from everywhere and at every time, creating a global scale marketplace of software services where providers offer their services interfaces and consumers may invoke them regardless geographical aspects, using the current Web infrastructure as the communication medium. Therefore, services are often implemented using standard Web-inspired languages and protocols and thus are called *Web Services*. Nowadays, Web Services are the technology commonly used when migrating legacy systems [1] to modern platforms or the technological protocol stack used for accessing information from smartphones [2].

Like any other software artifact, service interface descriptions have a size, complexity and quality, all of which can be measured [3]. In fact, previous research [4][5]

has emphasized on the importance of the non-functional concerns of services interfaces. Particularly, the work of [4] proposes a catalog of common bad practices found in services interfaces, which impact on the understandability and discoverability of the associated services. Understandability is the ability of a service interface description of being self-explanatory, which means a software engineer can reason about a service purpose just by looking at its interface description. Discoverability refers to the ability of a service of being easily retrieved, from a registry or repository, based on a partial description of its intended functionality such as a Google-like query. At the same time, in [3] the author describes a suite of metrics to assess the complexity and quality of services interfaces with respect to various aspects. Likewise, [5] proposes a suite comprising 4 metrics to assess service maintainability from service interface descriptions.

Methodologically, in practice service interfaces are not build by hand, but instead they are automatically generated by mapping programming languages constructors (and hence service implementations) onto service interface descriptions expressed in the Web Service Definition Language (WSDL). WSDL is an XML-based format for describing a service as a set of operations, which can be invoked via message exchange. In the Java arena, this mapping is usually achieved by tools such as Axis' Java2WSDL, Java2WS, EasyWSDL, and WSPProvide. The weak point of this methodology is that engineers partially control WSDL specification and therefore resulting WSDL descriptions may suffer from understandability, discoverability, complexity, quality and maintainability problems as measured by the aforementioned metrics catalogs.

We set forth the hypothesis that service developers can indirectly reduce the negative impact of some of these WSDL-level metrics by following certain programming guidelines at service implementation time. Particularly, in an attempt to explain the root causes of most *understandability* and *discoverability* problems associated with services interfaces, in [6] a statistical correlation analysis between service implementation metrics and service interface bad practices occurrences has been reported. In this paper, we study the feasibility of obtaining more *maintainable* services by exploiting Object-Oriented metrics (OO) values from the source code implementing services. Similarly to [6], the approach in this work uses OO metrics as early indicators to guide software developers towards obtaining more maintainable services.

Interestingly, we have found that there is a statistically significant, high correlation between several traditional (source code-level) OO metrics and the catalog of (WSDL-level) service metrics described in [5]. This is the most comprehensive and rigorously evaluated catalog of metrics for measuring service maintainability from WSDL interfaces. A corollary of this finding is that software developers could consider applying simple early code refactorings to avoid obtaining non-maintainable services upon generating service descriptions. Although our findings do not depend on the programming language in which services are implemented, we focus on Java, which is widely used in back-end and hence service development. To evaluate our approach, we performed experiments with a data-set of 154 real services, and the most popular Java-to-WSDL mapping tool, i.e. Java2WSDL (<http://ws.apache.org/axis/java>).

The rest of the paper is organized as explained next. Section 2 provides the background necessary to understand the goals and results of our research. Section 3 surveys related works. Section 4 presents detailed analytical experiments that evidence the

correlation of OO metrics with the Web Service metrics proposed in [5]. Section 5 explains how this correlation can be exploited to predict and early improve service maintainability. Section 6 concludes the paper and describes future research opportunities.

2 Basic Concepts

WSDL is a language that allows providers to describe two main aspects of a service, namely what it does (its functionality) and how to invoke it (its binding-related information). The former aspect reveals the functional service interface that is offered to potential consumers. The latter aspect specifies technological details, such as transport protocols and network addresses. Consumers use the former part to match third-party services against their needs, and the latter part to actually interact with the selected service. With WSDL, service functionality is described as a *port-type* $W = \{O_0(I_0, R_0), \dots, O_N(I_N, R_N)\}$, which lists one or more operations O_i that exchange input and return messages I_i and R_i , respectively. Port-types, operations and messages are labeled with unique names, and optionally they might contain some comments.

Messages consist of *parts* that transport data between providers and consumers of services, and vice-versa. Exchanged data is represented by using data-type definitions expressed in XML Schema Definition (XSD), a language to define the structure of an XML construct. XSD offers constructors for defining simple types (e.g. integer and string), restrictions, and both encapsulation and extension mechanisms to define complex constructs. XSD code might be included in a WSDL document using the *types* element, but alternatively it might be put into a separate file and imported from the WSDL document or external WSDL documents so as to achieve type reuse.

A requirement inherent to manually creating and manipulating WSDL and XSD definitions is that services are built in a *contract-first* manner, a methodology that encourages designers to first derive the WSDL interface of a service and then supply an implementation for it. However, the most used approach to build Web Services in the industry is *code-first*, which means that one first implements a service and then generates the corresponding service interface by automatically deriving the WSDL interface from the implemented code. This means that WSDL documents are not directly created by developers but are instead automatically derived via language-dependent tools. Such a tool performs a mapping T [6], formally $T : C \rightarrow W$.

T maps the main implementation class of a service ($C = \{M(I_0, R_0), \dots, M_N(I_N, R_N)\}$) to the WSDL document describing the service ($W = \{O_0(I_0, R_0), \dots, O_N(I_N, R_N)\}$). Then, T generates a WSDL document containing a port-type for the service implementation class, having as many operations O as public methods M the class defines. Moreover, each operation of W is associated with one input message I and another return message R , while each message comprises an XSD data-type representing the parameters of the corresponding class method. Tools like WSDL.exe, Java2WSDL, and gSOAP [7] rely on a mapping T for generating WSDLs from C#, Java and C++, respectively.

Fig. 1 shows the generation of a WSDL document using Java2WSDL. The mapping T in this case has associated each public method from the service code to an *operation* containing two *messages* in the WSDL document and these, in turn, are associated with an XSD data-type containing the parameters of that operation. Depending

on the tool used some minor differences between the generated WSDL documents may arise [6]. For instance, for the same service Java2WSDL generates only one port-type with all the operations of the Web Service, whereas WSDL.exe generates three port-types each bound to a different transport protocol.



Fig. 1. WSDL generation in Java through the Java2WSDL tool

The fact supporting our hypothesis is precisely that WSDL metrics in the general sense are associated with API design attributes [6,4]. These latter have been thoroughly studied by the software engineering community and as a result suites of OO class-level metrics exist, such as the Chindamber and Kemerer's catalog [8]. Consequently, these metrics tell providers about how a service implementation conforms to specific design attributes. For example, the CBO (Coupling Between Objects) metric gives a hint on data model quality in terms of code development and *maintenance* facilities. Moreover, the LCOM (Lack of Cohesion Methods) metric measures how well the methods of a class are semantically related to each other, or the *cohesion* design attribute.

An interesting approach is then to assess whether a desired design attribute as measured by an appropriate OO metric is ensured after WSDL generation as measured by a WSDL-level metric suitable for the attribute (e.g. [5] when targeting maintainability or [3] when targeting complexity). As a corollary, by using well-known software metrics on a service code C , a service developer might have an estimation of how the resulting WSDL document W will be like in terms of maintainability and complexity since a known mapping T deterministically relates C with W . Then, based on these code

metric/WSDL metric relationships, it is possible to determine a wider range of metric values for C so that T generates W without undesirable WSDL-level metric values.

3 Related Efforts

Although there has been a substantial amount of research to improve services interfaces quality [3,4,5], the approach to predict quality by basing on traditional OO metrics at development time remains rather unexplored. Indeed, this approach has been recently and exclusively explored in [6] to anticipate potential quality problems in services interfaces. Conceptually, the work presented in [6] studies the relationships between service implementation metrics and a key quality attribute of target services interfaces in WSDL, namely discoverability [4].

From services implementations the authors gathered 6 classic OO metrics, namely Chindamber and Kemerer's [8] CBO, LCOM, WMC (Weighted Methods Per Class), RFC (Response for Class), plus the CAM (Cohesion Among Methods of Class) metric from the work of Bansiya and Davis [9] and the well-known lines of code (LOC) metric. Additionally, they gathered 5 ad-hoc metrics, namely TPC (Total Parameter Count), APC (Average Parameter Count), ATC (Abstract Type Count), VTC (Void Type Count), and EPM (Empty Parameters Methods).

Regarding discoverability, the authors used a sub-set of the WSDL metrics listed in [4], which are focused on measuring aspects that affect discoverability, namely the legibility, conciseness, and understandability of WSDL descriptions [10,11,12]. Then, the authors collected a data-set of publicly available code-first Web Services projects, which by itself has been a valuable contribution to the field, and in turn analyzed the statistical relationship among metrics.

In the same direction that [6], this paper analyzes whether there are relations between service implementation metrics and the suite of metrics proposed by Baski and Misra [5], which comprises 4 novel metrics for measuring the complexity of the description of the information exchanged by Web Services. As will be explained later, these metrics can be statically computed from a service interface in WSDL, since this metric suite is purely based on WSDL and XSD schema elements occurrences.

3.1 Data Weight Metric

Baski and Misra [5] defined the data complexity as “the complexity of data flowed to and from the interfaces of a Web service and can be characterized by an effort required to understand the structures of the messages that are responsible for exchanging and conveying the data”. The definition of the Data Weight (DW) metric is based on the above, and computes the complexity of the data-types conveyed in services messages. To the sake of brevity, we will refer to the complexity of a message $C(m)$ as an indicator of the effort required to understand, extend, adapt, and test m , by basing on its structure. $C(m)$ counts how many elements, complex types, restrictions and simple types are exchanged by messages parts, as it is deeply explained in [5]. Formally:

$$DW(wSDL) = \sum_{i=1}^{n_m} C(m_i) \quad (1)$$

where n_m is the number of messages that the WSDL document exchanges. For the purposes of this paper, we have assumed n_m to consider only those messages that are linked to an offered operation of the WSDL document, thus it does not take into account dangling messages. The DM metric always returns a positive integer. The bigger the DM of a WSDL document, the more complex its operations messages are.

3.2 Distinct Message Ratio Metric

The Distinct Message Ratio (DMR) metric complements DW by attenuating the impact of having similar-structured messages within a WSDL document. As the number of similar-structured messages increases the complexity of a WSDL document decreases, since it is easier to understand similar-structured messages than that of various-structured ones as a result of gained familiarity with repetitive messages [5]. Formally:

$$DMR(wSDL) = \frac{DMC(wSDL)}{n_m} \quad (2)$$

where the Distinct Message Count (DMC) metric can be defined as the number of distinct-structured messages represented by $[C(m), n_{args}]$ pair, i.e. the complexity value $C(m)$ and total number of arguments n_{args} that the message contains [5]. The DMR metric always returns a number in the range of $[0,1]$, where 0 means that all defined messages are similar-structured, and 1 means that messages variety increases.

3.3 Message Entropy Metric

The Message Entropy (ME) metric exploits the probability of similar-structured messages to occur within a given WSDL document. Compared with the DMR metric, ME also bases on the fact that repetition of the same messages makes a developer more familiar with the WSDL document and results in ease of maintainability, but ME provides better differentiation among WSDL documents in terms of complexity. Formally:

$$ME(wSDL) = - \sum_{i=1}^{DMC(wSDL)} P(m_i) * \log_2 P(m_i) \quad (3)$$

$$P(m_i) = \frac{nom_i}{n_m}$$

where nom_i is the number of occurrences of the i^{th} message, and in turn $P(m_i)$ represents the probability that such a message occurs within the given WSDL document. The ME metric outputs values greater or equal than zero. A low ME value shows that the messages are consistent in structure, which means that data complexity of a WSDL document is lower than that of the others having equal DMR values.

3.4 Message Repetition Scale Metric

The Message Repetition Scale (MRS) metric analyses variety in structures of WSDL documents. By considering frequencies of $[C(m), n_{args}]$ pairs, MRS measures the consistency of messages as follows:

$$MRS(wSDL) = \sum_{i=1}^{DMC(wSDL)} \frac{nom_i^2}{n_m} \quad (4)$$

The possible values for MRS are in the range $1 \leq MRS \leq n_m$. When comparing two or more WSDL documents, a higher MRS and lower ME show that the developer makes less effort to understand the messages structures owing to the repetition of similar-structured messages.

4 Statistical Correlation among Services Metrics

We used the Spearman’s rank coefficient to analyze whether metrics taken on service implementations are correlated with metrics from their WSDL documents or not. Broadly, the experiment consisted on gathering OO metrics from real Web Services, calculating the service interface metrics of the previous section from WSDL documents, and analyzing the correlation among all pairs of metrics.

To perform the analysis, we employed a data-set of 154 WSDL documents from open source projects, which was the newest version available of the data-set described in [6] at the time of writing this article. All projects are written in Java, and were collected via the source code search engines Merobase, Exemplar and Google Code. Each project offers at least one Axis’ Java2WSDL Web Service description. Each service within each project consists of the implementation code and dependency libraries needed for compiling and generating WSDL documents. All in all, the generated data-set provided the means to perform a significant evaluation in the sense that the different Web Service implementations came from real-life software developers.

Regarding the correlation model, the independent variables were the WMC, CBO, RFC, LCOM, CAM, TPC, APC, ATC, VTC, EPM, and LOC metrics. On the other hand, the dependent variables were the DW, DMC, DMR, ME and MRS metrics. Metrics recollection is an extremely sensitive task for this experiment, but also a task that would require a huge amount of time to be manually carried on. Therefore, all the employed metrics have been automatically gathered by using an extended version of the *ckjm* [13] tool and a software library to automate the recollection of the WSDL metrics (<http://code.google.com/p/wSDL-metrics-soc-course/>).

Table 1. Correlation between OO metrics and WSDL ones

Metric	WMC	CBO	RFC	LCOM	LOC	CAM	TPC	APC	GTC	VTC	EPM
DW	0.47	0.80	0.47	0.47	0.47	-0.48	0.60	0.38	0.58	0.05	-0.03
DMC	0.82	0.53	0.82	0.82	0.82	-0.83	0.74	0.14	0.39	0.16	0.29
DMR	-0.71	0.22	-0.71	-0.71	-0.71	0.43	-0.37	0.34	0.25	-0.10	-0.36
ME	0.72	0.62	0.72	0.72	0.72	-0.79	0.68	0.18	0.45	0.13	0.18
MRS	0.80	-0.13	0.80	0.80	0.80	-0.54	0.49	-0.28	-0.18	0.10	0.37

Table 1 and Fig. 2 show the existing relations between the OO metrics (independent variables) and the WSDL metrics (dependent variables). For denoting those coefficients that are statistically significant at the 5% level or $p\text{-value} < 0.05$, which is a common choice when performing statistical studies [14], we employed bold numbers in Table 1 and circles in Fig. 2. The sign of the correlation coefficients (+, -) are depicted using black and white circles, respectively. A positive (black) relation means that when the independent variable grows, the dependent variable grows too, and when the independent variable falls the dependent goes down as well. Instead, a negative (white) relation means that when independent variables grow, dependent ones fall, and vice versa. The absolute value, or correlation factor, indicates the intensiveness of the relation regardless of its sign. Graphically, the diameter of a circle represents a correlation factor, i.e. the bigger the correlation factor the bigger the diameter in Fig. 2.

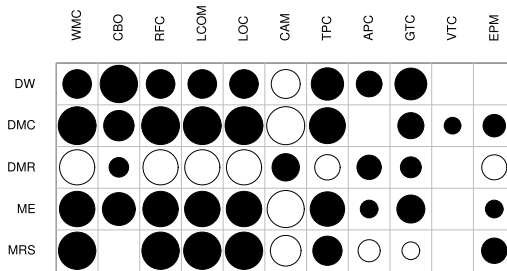


Fig. 2. Graphical representation of correlation analysis for rapidly identifying relations

5 A Step towards Early Improving WSDL Maintainability

By looking at Fig. 2 one could state that there is a high statistical correlation between the analyzed OO metrics and, at least, one metric for assessing the maintainability of services interfaces descriptions. Initially, this implies that every independent variable should be somehow “controlled” by software engineers attempting to obtain maintainable target WSDL documents. However, as determining the best set of *controllable* independent variables would deserve a deeper analysis within a longer paper, we will focus on determining a minimalist sub-set of OO metrics for this paper.

We employed two criteria for reducing the number of OO metrics. First, by basing on the study presented in [6] that shows the existence of groups of statistically dependent OO metrics, we select one representative metric for each group. In other words, if a group of variables in a data-set are strongly correlated, these variables are likely to measure the same underlying dimension (e.g. cohesion, complexity, or coupling). Table 2 shows the statistical correlation among OO metrics for the employed version of the data-set, which confirms that RFC, LCOM and LOC can be removed while keeping WMC, since these are statistically correlated at the 5% level with correlation factor of 1.

Table 2. Correlation among OO metrics

Metric	WMC	CBO	RFC	LCOM	LOC	CAM	TPC	APC	GTC	VTC	EPM
WMC	1.00	0.20	1.00	1.00	1.00	-0.84	0.76	-0.07	0.17	0.28	0.41
CBO	-	1.00	0.20	0.20	0.20	-0.37	0.29	0.26	0.41	-0.07	-0.15
RFC	-	-	1.00	1.00	1.00	-0.84	0.76	-0.07	0.17	0.28	0.41
LCOM	-	-	-	1.00	1.00	-0.84	0.76	-0.07	0.17	0.28	0.41
LOC	-	-	-	-	1.00	-0.84	0.76	-0.07	0.17	0.28	0.41
CAM	-	-	-	-	-	1.00	-0.63	0.08	-0.24	-0.35	-0.36
TPC	-	-	-	-	-	-	1.00	0.55	0.33	0.28	0.08
APC	-	-	-	-	-	-	-	1.00	0.30	0.04	-0.33
GTC	-	-	-	-	-	-	-	-	1.00	0.03	-0.18
VTC	-	-	-	-	-	-	-	-	-	1.00	0.38
EPM	-	-	-	-	-	-	-	-	-	-	1.00

Second, we removed the APC, GTC, VTC, and EPM metrics because they do not have at least one relationship with its correlation factor above $|0.6|$ at the 5% level. The rationale of this criterion reduction was to keep only the highest correlated pairs of variables.

Table 3. Minimalist sub-set of correlated OO and WSDL metrics

Metric	WMC	CBO	RFC	LCOM	LOC	CAM	TPC
DW	-	0.80	-	-	-	-	-0.74
DMC	0.82	-	0.82	0.82	0.82	-0.83	0.68
DMR	-0.71	-	-0.71	-0.71	-0.71	-	-0.60
ME	0.72	0.62	0.72	0.72	0.72	-0.79	-
MRS	0.80	-	0.80	0.80	0.80	-	-

Table 3 represents a minimalist sub-set of correlated metrics, and shows that the DW metric depends on two OO metrics, i.e. CBO and TPC. Then, the DW of a service may be influenced by the number of classes coupled to its implementation, and by the number of parameters their operations exchange. The results also show that DW is not highly influenced by cohesion related metrics, such as LCOM and CAM, neither by how many methods its implementation invokes (RFC) or methods complexity (WMC).

The DMC and ME metrics may be decreased by reducing the complexity of service implementation methods. This means that if a software developer modifies his/her service implementation and in turn this reduces the WMC, RFC and LOC metrics, such a

fall will cause a decrement in DMC and ME. At the same time, DMC and ME may be reduced by improving the cohesion of services implementations. This is because when the cohesion of services implementations is improved, LCOM falls and CAM rises, which may produce a lower value for DMC and ME, by basing on the signs of their respective statistical relations. ME is influenced by CBO as well.

The DMR metric has negative correlations with WMC, RFC, and LOC. Surprisingly, this means that when the complexity of services implementations methods grows, the ratio of distinct messages falls. Also, DMR has a positive correlation with TPC, which means that the higher the number of operations parameters the higher the ratio of distinct messages. The MRS metric presents high correlations with 3 complexity and 1 cohesion service implementation metrics.

The presented evidence shows that pursuing an improvement in the DMR metric may conflict with other metric-driven goals. This means that if a software developer modifies his/her service implementation and in turn this produces an increment in the WMC, RFC, LOC or LCOM metrics, such an increment will cause that DMC, ME, and MRS alert about an increment in the WSDL document complexity, however the DMR will fall. Clearly, incrementing DMC, and ME would be undesirable, but this could be the side-effect of a gaining in the DMR metric. As in general software literature, this kind of situations could be treated as *trade-offs*, in which the software engineer should analyze and select among different metric-driven implementation alternatives.

6 Conclusions

We have empirically shown that when developing code-first Web Services, there is a high statistical correlation between several traditional code-level OO metrics and some WSDL-related metrics that measure maintainability at the WSDL level. This enforces the findings reported in [6], in which a correlation between some OO metrics and metrics that measure service discoverability was also found.

As discussed in Section 5 this would allow software engineers and developers to early adjust OO metrics, for example via M. Fowler's code refactorings, so that resulting WSDLs and hence services are more maintainable. Moreover, modern IDEs provide specific refactorings that can be employed to adjust the value of metrics such as WMC, CBO and RFC. For metrics which are not very popular among developers and therefore have not associated a refactoring (e.g. CAM), indirect refactorings may be performed. For example, CAM is negatively correlated to WMC (see Table 2), and hence refactoring for WMC means indirectly refactoring for CAM.

At present, we are generalizing our results by analyzing correlation for WSDL documents built via code-first tools other than Java2WSDL. Besides, we are studying the correlation of OO metrics and the WSDL metrics proposed by Harry Sneed [3], which is a comprehensive catalog of metrics for measuring Web Service complexity.

We are also planning to deeply study the aforementioned trade-offs so as to provide more decision support to engineers. Indeed, refactoring for a particular OO metric may yield good results as measured by some WSDL metrics but bad results with respect to other WSDL metrics in the same catalog. Preliminary experiments with the WSDL complexity metrics catalog presented in [3] also suggest that reducing the value of some

OO metrics from Table 2 positively impacts on several complexity metrics, but at the same time negatively affects others. It is then necessary to determine to what extent these OO metrics are modified upon code refactoring so that a balance with respect to (ideally) all WSDL metrics in the catalog is achieved. This is however difficult specially when trying to balance the values of metrics of different catalogs, i.e. when attempting to build a service that is more maintainable according to Baski and Misra, but less complex according to Sneed.

Acknowledgments. We acknowledge the financial support provided by ANPCyT (PAE-PICT 2007-02311). We thank Martín Garriga for his predisposition and valuable help towards building the software tool that computes the studied Web Service maintainability metrics. We also thank Taiyun Wei, the author of the *R* script for drawing the correlation matrix of Fig. 2.

References

1. Rodriguez, J.M., Crasso, M., Mateos, C., Zunino, A., Campo, M.: Bottom-up and top-down COBOL system migration to Web Services: An experience report. *IEEE Internet Computing* (2011) (to appear)
2. Ortiz, G., De Prado, A.G.: Improving device-aware Web Services and their mobile clients through an aspect-oriented, model-driven approach. *Information and Software Technology* 52(10), 1080–1093 (2010)
3. Sneed, H.M.: Measuring Web Service interfaces. In: 12th IEEE International Symposium on Web Systems Evolution (WSE 2010), pp. 111–115 (September 2010)
4. Rodriguez, J.M., Crasso, M., Zunino, A., Campo, M.: Improving Web Service descriptions for effective service discovery. *Science of Computer Programming* 75(11), 1001–1021 (2010)
5. Baski, D., Misra, S.: Metrics suite for maintainability of extensible markup language Web Services. *IET Software* 5(3), 320–341 (2011)
6. Mateos, C., Crasso, M., Zunino, A., Coscia, J.L.O.: Detecting WSDL bad practices in code-first Web Services. *International Journal of Web and Grid Services* 7(4), 357–387 (2011)
7. Van Engelen, R.A., Gallivan, K.A.: The gsoap toolkit for web services and peer-to-peer computing networks. In: 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 128–135. IEEE Computer Society (2002)
8. Chidamber, S., Kemerer, C.: A metrics suite for Object Oriented design. *IEEE Transactions on Software Engineering* 20(6), 476–493 (1994)
9. Bansiya, J., Davis, C.G.: A hierarchical model for Object-Oriented design quality assessment. *IEEE Transactions on Software Engineering* 28, 4–17 (2002)
10. Fan, J., Kambhampati, S.: A snapshot of public Web Services. *SIGMOD Record* 34(1), 24–32 (2005)
11. Brian Blake, M., Nowlan, M.F.: Taming Web Services from the wild. *IEEE Internet Computing* 12, 62–69 (2008)
12. Pasley, J.: Avoid XML schema wildcards for Web Service interfaces. *IEEE Internet Computing* 10, 72–79 (2006)
13. Spinellis, D.: Tool writing: A forgotten art? *IEEE Software* 22, 9–11 (2005)
14. Stigler, S.: Fisher and the 5% level. *Chance* 21, 12–12 (2008)

Early Automated Verification of Tool Chain Design

Matthias Biehl

Embedded Control Systems
Royal Institute of Technology
Stockholm, Sweden
biehl@md.kth.se

Abstract. Tool chains are expected to increase the productivity of product development by providing automation and integration. If, however, the tool chain does not have the features required to support the product development process, it falls short of this expectation. Tool chains could reach their full potential if it could be ensured that the features of a tool chain are aligned with the product development process. As part of a systematic development approach for tool chains, we propose a verification method that measures the extent to which a tool chain design conforms to the product development process and identifies misalignments. The verification method can be used early in tool chain development, when it is relatively easy and cheap to perform the necessary corrections. Our verification method is automated, which allows for quick feedback and enables iterative design. We apply the proposed method on an industrial tool chain, where it is able to identify improvements to the design of the tool chain.

Keywords: Tool Integration, Process Modeling, Verification, Model-driven Development.

1 Introduction

Tool chains are becoming increasingly important in the development of complex systems, since tool chains have the potential to increase development productivity. In order to reach the goal of higher productivity, tool chains need to support the product development process. The development process of each product is different, specific to the company, its traditions, the preferences of the engineers and the set of development tools used. To provide support in development, a tool chain cannot be one-size-fits-all solution, but it needs to be customized to the development context it is used in [7]. It is especially important that the tool chain is customized and aligned with the product development process, otherwise practitioners do not accept a new tool chain in their work [5]. It is thus important to verify that a tool chain is suitable and fits in a specific development context. We verify by checking the degree to which a tool chain is aligned with a given product development process.

This work is part of a vision to support the development of a tool chain using a systematic methodology and tools that manage the complexity of tool chain development [3]. As part of the systematic development methodology, the design of a customized tool chain needs to be analyzed and verified early in the tool chain development process[4]. When verifying the design of a tool chain, we are interested in checking how well the design realizes the requirements. A verification of the design is a complement - not a replacement - to a verification of the implementation. Early verification of tool chain design can detect possible misalignments between the product development process and the tool chain, when corrections are still relatively simple and cheap. In addition, when we automate the early verification of tool chain design, it can be performed repeatedly with little effort, allowing for iterative development of tool chains. This leads to the central research question addressed in this paper: *How can early verification of the alignment between tool chain design and product development process be performed and automated?*

The remainder of this paper is structured as follows. In section 2 we describe the approach for automated verification. In section 3 we investigate the formal modeling of the product development process, the tool chain and the relation between these models. In section 4 we devise a method to identify misalignments between the development process and the design of the tool chain and we measure the degree of alignment in section 5. In section 6 we apply the verification method on an industrial tool chain and discover possible improvements. In sections 7 and 8 we relate our approach to other work in the field, list future work and consider the implications of this work.

2 Approach

Our goal is to support the development of tool chains by verifying that the tool chain design is aligned to the product development process. The verification produces a list of misalignments and a measurement indicating the degree of alignment between the tool chain and the product development process.

In figure 1 we give an overview of the verification approach. As input we need a formalized description of the tool chain design (section 3.2) and a description of the process including the set of tools and their capabilities (section 3.1). These descriptions need to be in an appropriate language for describing the tool chain and its needs. An initial verification graph is created based on the two input models. We automatically add mapping links to the verification graph, which are constructed according to mapping rules (section 3.3). We then apply alignment rules (section 4) on the verification graph to find out in how far the tool chain supports the development process. We also apply metrics (section 5) to determine the degree of alignment between the tool chain and the process.

¹ Note: In this paper we deal with two different development processes: the development process of the tool chain, called *tool chain development process*, and the development process of some product, where the development process will be supported by the tool chain (see figure 4), called *product development process*.

The metrics and the list of misalignments provide feedback for an iterative design and guide the tool chain designer towards building an improved, more useful tool chain that is aligned with the process.

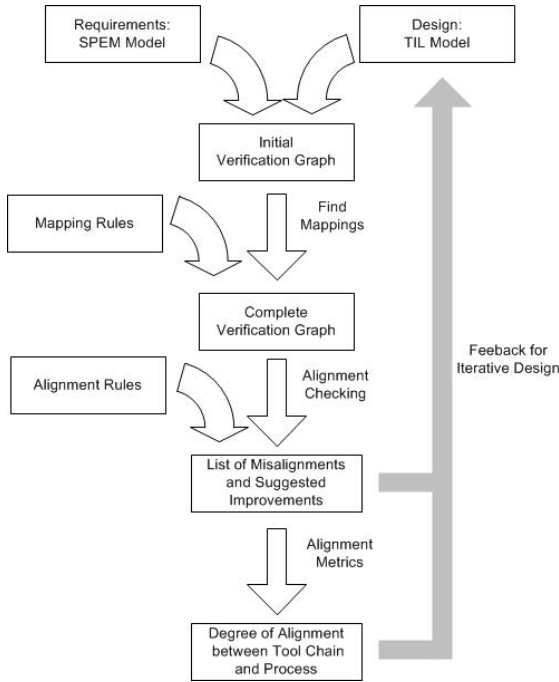


Fig. 1. Approach for early automated verification of tool chains

3 Building Up the Verification Graph

In this section, we describe how we build up the data structure used for verification. The data structure combines the description of the development process (section 3.1) and the tool chain design (section 3.2) by adding mapping links between both descriptions (section 3.3).

3.1 Description of the Product Development Process

We apply the Software & Systems Process Engineering Metamodel (SPEM) [10] to describe both the product development process and the set of tools used. A SPEM model can thus be used to describe the process requirements of a tool chain. A number of concepts are defined in SPEM, we focus here on the core concepts relevant in this context. A *Process* is composed of several *Activities*.

An *Activity* is described by a set of linked *Tasks*, *WorkProducts* and *Roles*. A *Role* can $\ll\text{perform}\gg$ a *Task* and a *WorkProduct* can be marked as the $\ll\text{input}\gg$ or $\ll\text{output}\gg$ of a *Task*. A *WorkProduct* can be $\ll\text{managed by}\gg$ a *Tool* and a *Task* can $\ll\text{use}\gg$ a *Tool*. An example model is provided in figure 4.

SPEM can be used at several levels of abstraction: A high-level description may consist only of a process and activities and a low level description may break down each task into steps and detailed guidances. In the following we assume a process description on an appropriate level of abstraction that contains at least instances of all of the above metaclasses.

3.2 Description of the Design of the Tool Chain

We would like to create an early design model that describes all important design decisions of a tool chain. We chose to apply the Tool Integration Language (TIL) 3, a domain specific modeling language for tool chains. TIL allows us not only to model a tool chain, but also to analyze it and generate code from it. Here we can only give a short overview: In TIL we describe the tool chain in terms of a number of *ToolAdapters* and the relation between them. A *ToolAdapter* exposes data and functionality of a tool. The relation between the *ToolAdapters* is realized as any of the following *Channels*: a *ControlChannel* describes a service call, a *DataChannel* describes data exchange and a *TraceChannel* describes the creation of trace links. An example for using the graphical language of TIL is provided in figure 5.

3.3 Mapping Rules and the Verification Graph

For the purpose of analysis we create an *initial verification graph*, which contains all the elements and internal links from both the SPEM model and the TIL model. We assume that the descriptions of the process and of the tool chain are internally verified and consistent, i.e. that all used variables have been defined. This graph is the basis for the construction of mapping links that connect model elements from the SPEM and TIL model. The mappings are constructed according to the mapping rules defined in table 1.

Table 1. Mapping rules for SPEM and TIL metaclasses

SPEM Metaclass	TIL Metaclass
RoleDefinition	User
ToolDefinition	ToolAdapter
TaskDefinition	Channel

The mapping rules introduce mapping links between the model elements of the given types that also have similar instance names. We measure the similarity of names by using the Levenstein distance 9. This ensures that corresponding elements will be mapped, even if they do not have the exact same name: e.g. a

ToolDefinition in SPEM with name 'UML Tool' will be mapped to a *ToolAdapter* in TIL with name 'Papyrus UML TOOL'. We add the mapping links to the initial verification graph, which results in a *complete verification graph* exemplified in figure 2.

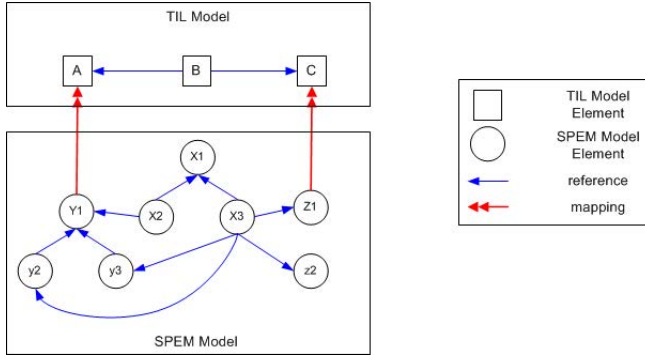


Fig. 2. Example of a complete verification graph

4 Verification by Alignment Checking

In order to verify a given tool chain design, we check for alignment of the requirements provided by a SPEM model with the design provided by a TIL model. We base the alignment check on the verification graph described in the previous section and a number of alignment rules.

4.1 Alignment Rules

The basic assumption of our alignment rules is that each SPEM *TaskDefinition* that has two different SPEM *ToolDefinitions* associated with it, is a candidate task for support by the tool chain; we call it integration-related task. Arbitrary SPEM *TaskDefinitions* connected to only one tool, are of no interest for tool integration; for these tasks, users will work directly with the GUI of a single tool. We distinguish two alignment rules in figure 3: (A) for tool-oriented *TaskDefinitions* and (B) for *WorkProducts*.

(A) For each *SPEM TaskDefinition* with two *SPEM ToolDefinitions* associated with it, this rule expects corresponding *TIL ToolAdapters* and a *TIL Channel* between them. A pair of *TIL ToolAdapters* and *SPEM ToolDefinition* are corresponding if they have mapping links between them (see section 3.3). This rule requires two such pairs, one pair with name X and one pair with name Y, where $X \neq Y$. Optionally, for each *SPEM RoleDefinition* that is connected to the *SPEM TaskDefinition*, we expect a corresponding *TIL User* that is connected via a *TIL ControlChannel* with the *TIL Channel*.

(B) For each *SPEM TaskDefinition* that has associated *SPEM WorkProducts* as input and output, where the input *SPEM WorkProduct* has a different associated *SPEM ToolDefinition* than the output *SPEM WorkProduct*, the rule expects to find *TIL ToolAdapters* corresponding to the *SPEM ToolDefinition* and a *TIL Channel* between them.

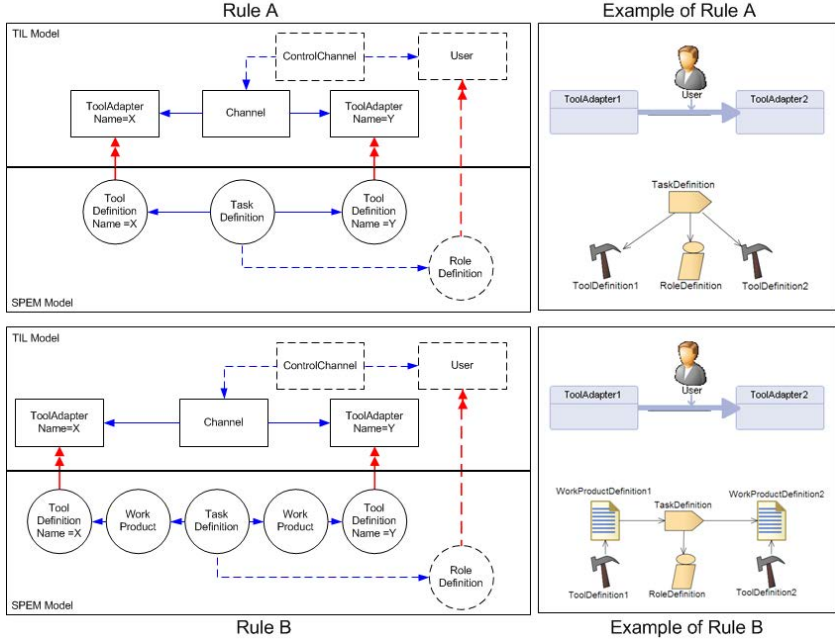


Fig. 3. Alignment rules and examples: (A) for tool-oriented tasks and (B) for work products, using the notation of figure 2

The rules contain optional parts, visualized by dashed elements. For both rules A and B we need to check the (1) realized and (2) necessary alignments, thus the alignment rules can be interpreted in two directions:

- (1) If the lower part of the rule can be found in the SPEM model, the upper part needs to be found in the TIL model as well. Each integration-related task in the process model should also be represented in the tool chain, we call it a *realized alignment*, because the requirements provided by the process model are realized by the tool chain. This direction of rule interpretation checks if the requirements given by the product development process are actually realized by the tool chain.
- (2) If the upper part of the rule can be found in the TIL model, the lower part needs to be found in the SPEM model. Each *Channel* in the tool chain design should have a corresponding requirement in the process model, we call it a *necessary alignment*, because this part of the tool chain is necessary for

fulfilling the requirements of the process This direction of rule interpretation checks if the tool chain contains superfluous features that are not used by the process.

4.2 Alignment Checking

We perform pattern matching of the alignment rules on the verification graph. As a result of the pattern matching we get a list of alignments and a list of misalignments. An alignment is a match of an alignment rule, a misalignment is a mismatches of the alignment rules, where only the upper part (TIL part) or lower part (SPEM part) of the rule can be found in the verification graph. Assuming that the product development process is described correctly, we can add a recommendation for resolving the misalignment by adapting the tool chain design in TIL. If a *Channel* is missing, we can suggest to add it to the TIL model because it would improve the functionality of the tool chain. If the *TaskDefinition* is missing in the SPEM model, we can suggest to remove the *Channel* from the TIL model, because it is not required.

5 Quantification by Alignment Metrics

Tool chains provide increased automation of integration-related tasks. Moreover, the automation needs to be appropriate to the needs of the tool chain. Appropriate automation depends on the alignment of tool chain and product development process, the set of product development tools and their capabilities. We measure usefulness based on two degrees of alignment.

The degree of alignment can be determined based on the results of the alignment checks explained in the previous section. We make the following definitions:

- $\#matches_{SPEM}$ are the number of matches of the SPEM part (lower part) of rules A and B.
- $\#matches_{TIL}$ are the number of matches of the TIL part (upper part) of rules A and B.
- $\#matches$ are the number of matches of the combined SPEM and TIL part of rules A and B.

The recall alignment in formula 1 measures the ratio of integration related tasks in the process model that are realized by a channel in the tool chain. The precision alignment in formula 2 measures, the ratio of channels in the tool chain are required by a corresponding integration-related task in the process model.

$$alignment_{recall} = \frac{\#matches}{\#matches_{SPEM}} \quad (1)$$

$$alignment_{precision} = \frac{\#matches}{\#matches_{TIL}} \quad (2)$$

A well-aligned tool chain has both a high degree of precision alignment and a high degree of recall alignment, typically expressed by F-measure. A high degree of precision alignment shows that the tool chain has little or no superfluous

features. A high degree of recall alignment shows that the tool chain has all or almost all the features needed. A low degree of precision alignment can be fixed by reducing unused features from the tool chain. These features will not be used according to the process description and only increase the development cost of the tool chain. A low degree of recall alignment might be even more critical. It needs to be fixed by introducing new features into the tool chain design, so the integration-related tasks are supported by the tool chain.

6 Case Study

This case-study shows the development process of an embedded system that is characterized by tightly coupled hardware and software components:

- The requirements of the system are captured in a requirements management tool. The system architect designs a UML component diagram and creates trace links between UML components and the requirements.
- The UML model is refined and a fault tree analysis is performed in the safety analysis tool. When the results are satisfactory, the control engineer creates a Simulink model for simulation and partitions the functionality for realization in software and hardware.
- The application engineer uses Simulink to generate C code, which is refined in the WindRiver tool. The data from UML and Simulink is input to the IEC-61131-3 conform ControlBuilder tool. The data from ControlBuilder, Simulink and WindRiver is integrated in the Freescale development tool for compiling and linking to a binary for the target hardware.
- A hardware engineer generates VHDL code from Simulink and refines it in the Xilinx ISE tool.

A SPEM model of this development process for hardware-software co-design used by the company is modeled as shown in figure 4. The tool chain designer creates the initial tool chain design shown in figure 5. Both models are input to our algorithm, which identifies the misalignments shown in table 2.

As it turns out, the tool chain designer built the tool chain according to the verbatim description of the process given above. This description, however, does not cover the intricate details of the relationships between tools that are present in the SPEM model. The tool chain design only contains forward links between tools, as they would be used in a waterfall process model, but neglects the cross links and dependencies typical for modern engineering processes. In this case study we have a cross link between Simulink and ControlBuilder and one between WindRiver and ControlBuilder. In addition, the process model specifies a pair of links between ControlBuilder and FreeScale and another pair between Simulink and FreeScale, but the tool chain design only realizes one link out of each pair.

For the case study the degree of recall is 69%, the degree of precision is 100%. This tells us that the tool chain has no superfluous features, but does not realize all necessary ones. The tool chain could be improved by adding the four missing *Channels* to the TIL model, as listed in table 2.

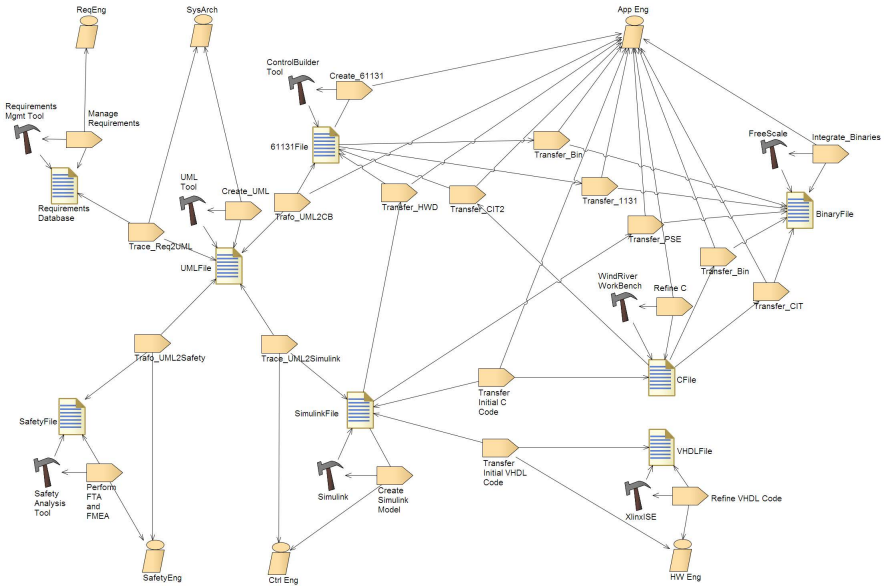


Fig. 4. Product development process of the case study as a SPEM model

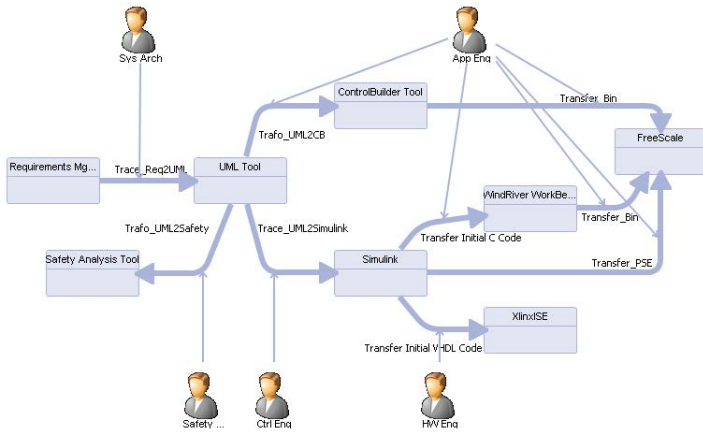


Fig. 5. Tool chain of the case study as a TIL model

Table 2. The misalignments and suggested improvements for the tool chain design

TaskDefinition	Input ToolDefinition	Output ToolDefinition	Suggested Improvement
Transfer_HWD	Simulink	ControlBuilder Tool	Add TIL Channel
Transfer_CIT2	WindRiver WorkBench	ControlBuilder Tool	Add TIL Channel
Transfer_1131	ControlBuilder Tool	FreeScale	Add TIL Channel
Transfer_CIT	Simulink Tool	FreeScale	Add TIL Channel

7 Related Work

There are a number of approaches for tool integration, as documented in the annotated bibliographies [4,12], however, most tool integration approaches do not explicitly take the development process into account. In this section we focus on the few approaches that acknowledge a relationship between process and tool chain. We classify this related work on tool integration according to two dimensions: Executing the process using (1a) interpretation vs. (1b) compilation and applying a process modeling formalisms that is (2a) proprietary vs. (2b) standardized.

(1a) Interpretation-based approaches use the process definition directly to integrate tools; this techniques is also known as enactment of process models. Since the description of the process is the same as that of the tool chain, the two have the advantage to always be aligned. There are two preconditions for the interpretation approach: the process model needs to be executable and the access to data and functionality of the development tools needs to be possible.

(2a) The use of a proprietary process model in tool chains is introduced in [6] as the process-flow pattern. In this approach a workflow realizes control integration and is executed by interpretation. (2b) A well known standardized process model is SPEM. SPEM itself is not executable but there are extensions to SPEM [10] that make the process model executable [8,2]. The orchestration of tools by a process model is shown in [11]. However, the interpretation of integration related tasks is often not possible, since the interfaces to the development tools are too different. Thus the use of process enactment to build tool chains is limited.

(1b) Compilation based approaches transform the process model into another format, where the process model serves as a set of requirements. (2a) Proprietary process models provide great flexibility to adapt them to the specific needs of tool integration. An integration process model is developed in [1], where each process step can be linked to a dedicated activity in a tool. For execution it is compiled into a low-level process model. The proprietary process model needs to be created specifically for constructing a tool chain. (2b) In this work we assume a compilation-based approach, where a process model is compiled or manually transformed into a tool chain design. We use the standardized process metamodel SPEM [10], which allows us to reuse existing process models as a starting point for creating a tool chain design. The described approach verifies that a tool chain design fulfills the requirements described in SPEM.

8 Future Work and Conclusion

In this work we propose a verification method for early design that checks the alignment of a tool chain with a product development process. We formalize this relationship, which allows us to reason about the relationship in an automated way and verify the tool chain against the needs of the product development process using consistency checks and metrics.

The verification rules presented in section 4 may not only be used analytically for verification, but also constructively for automatically building a design

model for a tool chain from a given process description. In the future we would like to explore this idea and integrate it into a comprehensive methodology for developing tool chains. We would also like to apply this approach on several bigger case studies. The case study should contain a thorough investigation of the false positives and false negatives found with the approach.

The approach contributes to the vision of a systematic development methodology for tool chains with a potential for cost savings and an increased time to market. An important prerequisite for reaching this potential is a clear description of the product development process that the tool chain is intended to support. The approach guides the tool chain designer by analyzing different integration options and eventually building a tailored tool chain that is suitable and useful to the tool chain user in the context of a given product development process.

References

1. Balogh, A., Bergmann, G., Csertán, G., Gönczy, L., Horváth, Á., Majzik, I., Pataricza, A., Polgár, B., Ráth, I., Varró, D., Varró, G.: Workflow-Driven Tool Integration Using Model Transformations. In: Engels, G., Lewerentz, C., Schäfer, W., Schürr, A., Westfechtel, B. (eds.) *Nagl Festschrift. LNCS*, vol. 5765, pp. 224–248. Springer, Heidelberg (2010)
2. Bendraou, R., Combemale, B., Cregut, X., Gervais, M.P.: Definition of an Executable SPEM 2.0. In: 14th Asia-Pacific Software Engineering Conference, APSEC 2007, pp. 390–397. IEEE (December 2007)
3. Biehl, M., El-Khoury, J., Loiret, F., Törngren, M.: A Domain Specific Language for Generating Tool Integration Solutions. In: 4th Workshop on Model-Driven Tool & Process Integration (MDTPI 2011) (June 2011)
4. Brown, A.W., Penedo, M.H.: An annotated bibliography on integration in software engineering environments. *SIGSOFT Notes* 17(3), 47–55 (1992)
5. Christie, A., et al.: Software Process Automation: Interviews, Survey, and Workshop Results. Technical report, SEI (1997)
6. Karsai, G., Lang, A., Neema, S.: Design patterns for open tool integration. *Software and Systems Modeling* 4(2), 157–170 (2005)
7. Kiper, J.D.: A framework for characterization of the degree of integration of software tools. *Journal of Systems Integration* 4(1), 5–32 (1994)
8. Koudri, A., Champeau, J.: MODAL: A SPEM Extension to Improve Co-design Process Models. In: Münch, J., Yang, Y., Schäfer, W. (eds.) *ICSP 2010. LNCS*, vol. 6195, pp. 248–259. Springer, Heidelberg (2010)
9. Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Doklady Akademii Nauk SSSR* 163(4), 845–848 (1965)
10. OMG. Software & Systems Process Engineering Metamodel Specification (SPEM). Technical report, OMG (April 2008)
11. Polgar, B., Rath, I., Szatmari, Z., Horvath, A., Majzik, I.: Model-based Integration, Execution and Certification of Development Tool-chains. In: Workshop on Model Driven Tool and Process Integration (June 2009)
12. Wicks, M.N.: Tool Integration within Software Engineering Environments: An Annotated Bibliography. Technical report, Heriot-Watt University (2006)

Using UML Stereotypes to Support the Requirement Engineering: A Case Study

Vitor A. Batista, Daniela C.C. Peixoto, Wilson Pádua, and Clarindo Isaías P.S. Pádua

Computer Science Dept., Federal University of Minas Gerais, Brazil
{vitor,cascini,wilson,clarindo}@dcc.ufmg.br

Abstract. In this paper we discuss the transition of an educational process to real-life use. Specifically, a Requirements Engineering (RE) process was tailored and improved to comply with the organization business goals. We discuss challenges faced and proposed solutions, focusing on automation and integration support for RE activities. We use stereotypes to enhance UML diagram clarity, to store additional element properties, and to develop automated RE process support. Stereotypes are one of the core extension mechanisms of the Unified Modeling Language (UML). The benefits found in their use in a software development organization support the claims that stereotypes play a significant role in model comprehension, reduce errors and increase productivity during the software development cycle.

Keywords: Stereotype, Unified Modeling Language, User Interface Prototyping, Functional Point, Constraints.

1 Introduction

Requirement Engineering (RE) is a very labor-intensive and critical process. It is by nature a manual activity, since requirements are informally elicited by analysts from users, and represented in abstract specifications. It is a critical activity because errors inevitably lead to later problems in design and implementation, which, usually, are expensive and difficult to repair. Many studies have shown that project failures often result from poor quality requirements [1]. It is almost impossible to keep delivery times, costs and product quality under control if the requirements are not adequately specified and managed [2]. Baziuk [3] showed that it may cost up to 880 times more expensive to detect and repair error in the maintenance stage, compared to detecting and repairing them during the RE phase. In order to produce software that closely matches the stakeholders' needs, great attention should be paid to RE activities.

One way to help analysts is to provide some automated and integrated functionalities for RE. This helps to reduce manual labor, and allows the early detection of errors. Indeed, as a naturally manual labor-intensive process, only a few functionalities can be automated. However, their impact on rework reduction and increasing productivity can be substantial.

In this work, we present the adoption and improvement of an RE process in a software development organization, Synergia. It adopts a professional version of Praxis (an educational software development process), called Praxis-Synergia [4]. In this context, we discuss some challenges faced and proposed solutions, which include an automated and integrated approach for carrying out RE activities.

Our main contribution lies in the automation approach itself and in the subsequent steps in the RE activities, such as, automatic prototype generation, support for counting functional size and model verification. These facilities are integrated using a UML Profile with several stereotypes. The idea of stereotypes usage came from the educational version of the Praxis process, which applies stereotypes mainly to enhance UML diagram clarity and to store additional element properties, eliminating the need for many text and spreadsheet model attachments. In the professional version of this process, used at Synergia, an additional goal was to extend the usage of stereotypes, developing automated functionalities to support RE. The expected benefits were improving model comprehension, reducing errors and increasing.

The article is structured as followed. Section 2 introduces basic concepts of Praxis and Section 3 presents the details of its requirement engineering activities. Section 4 provides the application of these methods in a software development organization, explaining the problems faced and solutions proposed. Section 5 provides a discussion of the results and open issues. Section 6 presents the related work. Section 7 concludes and presents future work.

2 Background

Praxis is a model-based, use-case-driven process. In its prescribed development sequence, a project is divided in iterations, where each iteration implements one or more functional requirements, modeled as use cases.

Praxis models the development of each use case workflow as a sequence of development states. The name of each state evokes how far the use case has advanced towards complete implementation and validation. Each state is associated with a set of resulting work products [5], which become inputs to the following steps.

The requirements set is captured in a UML **Problem model**, containing a requirements view, that expresses the users' viewpoint, and an analysis view, the developers' vision of the problem. The design is captured in a UML **Solution model**; in its main views, a use view shows how the user will view the proposed system; a test view shows the design of the system tests; and a logical view shows the system innards at a high level. Both product and test code are derived from the Solution model, which may be partially reverse-engineered from the code, for documentation purposes.

In its original form, Praxis has been mainly used for educational purposes [6]. A tailored version has been used for professional software development in the authors' laboratory, Synergia, as described in [4]. This paper focuses on the application of the professional version.

The Praxis distributed material includes, among other artifacts, a stereotype library that provides UML stereotypes employed both by the Problem and Solution model, at

the reusable framework level and at the specific application level. Those stereotypes provide a number of services. This library is contained in a **Profile model**. In the professional version, this profile is deployed within a modeling tool plug-in, while it is directly accessed by the modeler, in the educational version. Currently, the modeling tool is the IBM Rational modeling toolset, in its Modeler or Architect versions (RSM and RSA respectively).

3 Standard Requirement Engineering Process Version

In this section we summarized Requirements and Analysis processes of the Praxis standard educational version. As described above, the Problem model captures and analyses requirements in two main views, Requirement and Analysis; its structure closely follows the IEEE-830 standard for Requirements Specification.

The Requirement view represents desired functions as UML Use Cases, stereotyped as «appUseCase». Each Use Case behavior is described by one or more event flows (scenarios), classified as:

- **Main flow:** represents the most usual scenario in the use case execution.
- **Alternate flows:** represent less usual, optional or exceptional scenarios.
- **Subflows:** represent sequences of steps invoked in the other kinds of flows.

Each of these types of flows is modeled as a stereotyped UML Activity, with «mainFlow», «altFlow» and «subFlow» stereotypes, respectively. Figure 1 shows a use case with several flows (Activities) and a diagram of its Main flow.

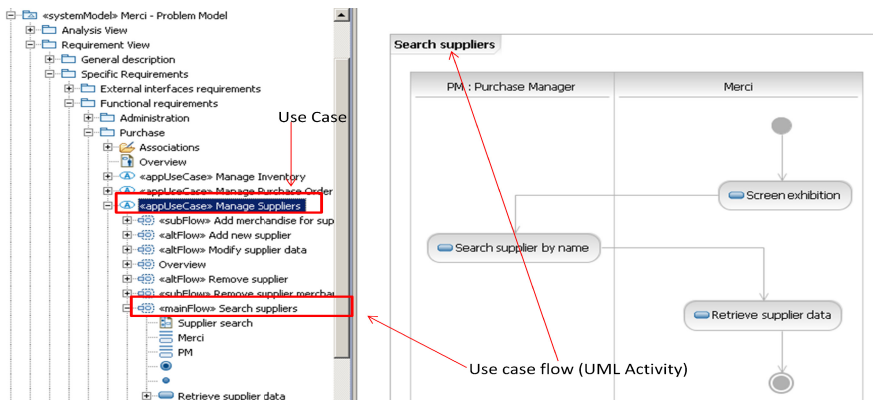


Fig. 1. Use case flow modeled as an Activity

The Analysis view describes the product as seen by the developer, but still at abstract problem level; it models concepts, procedures and interfaces as classes, and their interactions by stereotyped UML collaborations.

These realize, in conceptual terms, the functionality described by the use cases, using UML Interactions, usually represented by Sequence Diagrams. As a general

modeling rule, each scenario is realized by an interaction with the same stereotype. Collaboration attributes represent instances of the participating analysis classes. These classes apply Jacobson's standard stereotypes «entity», «control» and «boundary». A fourth stereotype, «persistentEntity», was created to tag data that should or should not be persisted by application. User's interfaces are prototyped using MS Office, and the prototypes are attached to «boundary» classes.

4 Professional Requirement Engineering Process Version

Synergia, the organization where this study was carried out, is a laboratory for software and systems engineering, hosted by the Computer Science Department at the Federal University of Minas Gerais. It has some 75 people in its staff, composed by undergraduate and graduate students and non-student graduates, mostly with a computer science background. Both standard and professional versions of Praxis have been recently upgraded to version 3.

During the upgrade of Praxis-Synergia from version 2 to 3, some issues and problems arose, mainly regarding the execution of manual activities that could be automated and the storage of additional information. In the next sections we describe each problem faced and the solution provided, focusing on the RE process. In section 5, we discuss the achieved benefits.

The core idea of the proposed solutions is extend the original Praxis UML Profile adding several stereotypes more, tagged values and OCL Constraints. We also developed some automated functions inside the IBM Rational Software Architect (RSA) plug-in, to support the engineers' work.

4.1 Automatic Prototype Generation

As we discussed above, the user interfaces in the Praxis standard version are modeled as classes with «boundary» stereotype and prototyped using MS Office. Besides providing the conventional boundary shape, the «boundary» classes record IEEE-830 recommended data, such as information source and destination, format and kind. Each required user command and interface field are modeled using stereotyped «reqOperation» and «reqAttribute» attributes respectively, which record, for example, valid conditions of the operation, pertinence to (visually distinct) groups, content types (text, integers, real numbers, enumerations etc.), edition requirements and other problem-level information. During the first use of the process in actual projects, we identified the following issues:

- A large amount of fields and commands were used, which made the reading and understanding of the UML model difficult. In addition, field name prefixes were used to represent related groups.
- Changes in user interface requirements needed changes in two artifacts: the UML classes and the prototype. Frequently, they became inconsistent.

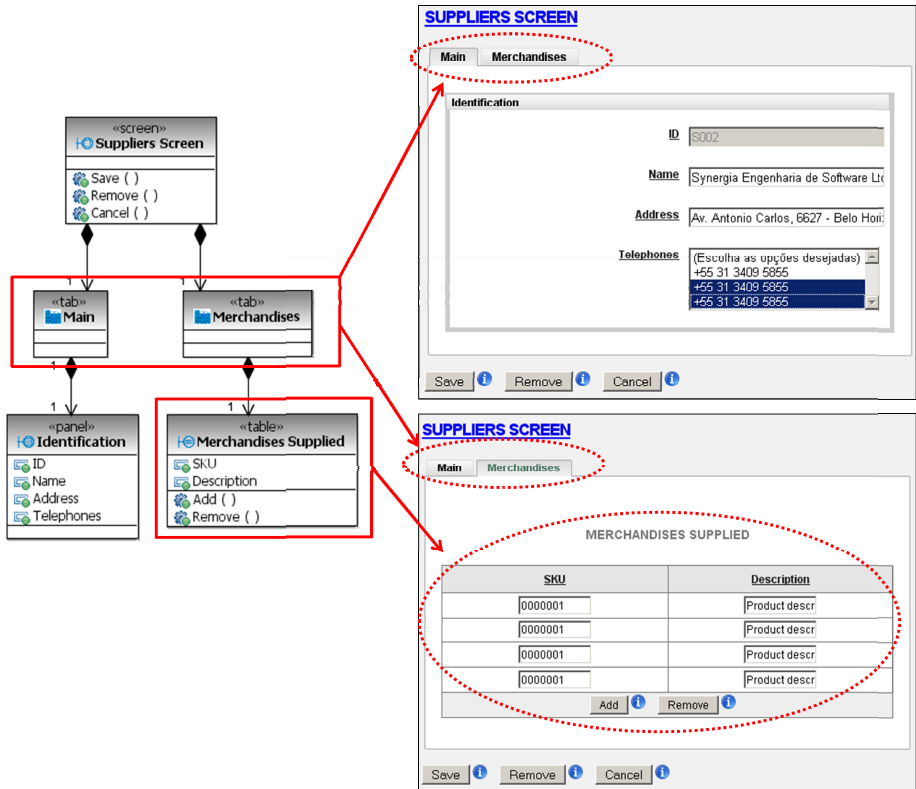


Fig. 2. The stereotypes classes used to model a screen, and their respective prototypes

In order to solve these issues, we extended the Praxis stereotypes library, adding new class stereotypes to represent user interfaces and their components. With this new information, it was possible to automate the generation of an HTML prototype, consistent by construction with the UML model.

The list of stereotypes used to elaborate the prototype includes: «screen», «panel», «table», «tab» and «report» for classes, «field» for their attributes and «command» for their operations. UML composition associations between classes indicate how stereotyped interface components are integrated with their parent's component in the prototype screen. Figure 2 shows the UML classes used to represent a screen, and its corresponding generated prototype. Note the use of «tab», «panel» and «table» stereotypes in the class model and in the prototype.

Navigation behavior is modeled using directed associations with «navigate» stereotype which stores a list of «command» operations that causes this navigation to occur. The HTML prototype is generated using this information to provide navigation through JavaScript code.

Automatic prototype generation also uses attribute data from «field» stereotypes to add visual information in a popup Windows. Information about mandatory, valid values, format, enablement and business rules associated are displayed.

In order to improve model understanding, users can access the complete specification of the attributes by clicking on their names on prototypes.

4.2 Counting Function Points

Function Point (FP) counting is a major Synergia concern, since most of its contracts are based on FP counts, and this information is crucial to measure productivity in FP per person-month, to be used in the estimation of future projects. We use Unadjusted Function Points (UFP), since Adjusted Function Points are not standardized by ISO, and the Praxis process has other means to compute the cost of non-functional requirements.

Use case points (UCP) are sometimes considered as size proxies closer to UML. Besides the customary use of FP by our clients, we did not use them because they have no official standardization, such as IFPUG and ISO provide for FP; and because different levels of use case description, such as superordinate vs. subordinate, may yield very different counts.

Standard Praxis records FP counting in spreadsheets of the MS Excel. Using this artifact format brought some specific difficulties to Synergia, because requirements changes are very common among some of our clients. Monitoring size evolution, by recording continuously updated FP counts, was an error-prone manual procedure. Besides, since FP counting was based on the **Problem model** and recorded in another artifact, that procedure often caused inconsistencies.

One solution was to count FP directly in the UML model, storing the information on the stereotypes attributes. This minimizes inconsistencies caused by unavoidable requirements changes, and allows monitoring project size evolution while its requirements are still being elicited. The following sections describe the procedure used to counting the data and transactional FP. They are discussed in more detail in [30], but are summarized here for the sake of completeness.

Counting Data Functions

The Problem model uses UML classes stereotyped with «persistentEntity» to represent persistent data. Since the FP counting procedure uses two different kinds of Data Functions, these are modeled using two new stereotype specializations:

1. ILF (*Internal Logical File*), corresponding to data that are maintained by the application under development, are modeled by the «internalPersistentEntity» stereotype.
2. EIF (*External Interface File*), corresponding to data that are maintained by other applications, but read by the application under development, are modeled by the «externalPersistentEntity» stereotype.

Figure 3 shows the metamodel for those stereotypes. The original educational Praxis «persistentEntity» becomes an abstract stereotype, holding common properties that are useful in the FP counting method.

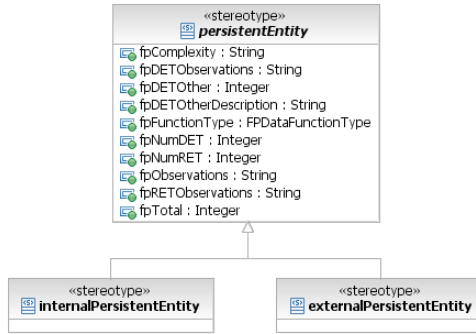


Fig. 3. Stereotypes for counting Data Functions

The mapping between the persistent entities and the Data Functions is neither direct nor one-to-one. Automating this mapping is quite complex, and, therefore, in our method, must be manually performed. The FP counting specialist must decide how persistent classes should be grouped, in order to be mapped onto Data Functions (for those not familiar with FP, this is also a standard procedure in manual counting). Figure 4 illustrates a model with four classes and their proposed grouping, which led the specialist to count two ILFs. In Data Functions groups with more than one class, one of them is chosen to stand for the group (in our example, Purchase Order). After grouping and mapping, the attributes in Table 1 should be filled in each main class.

To support this procedure, the RSA plug-in eases the filling of the stereotype properties and calculates the complexity and total function points for each Data Function. Figure 5 shows an example where the RSA plug-in counts the ILF Purchase Order, mapped to classes Purchase Order and Purchase Order Item.

In the example shown in Figure 4, the list of manually grouped classes is selected among the classes which are not yet assigned to Data Functions, and corresponding number of RETs is manually informed. After that, the user may select any attribute of the grouped classes (including association attributes) as DETs. Additionally, other DETs, not explicitly derived from attributes (such as messages and commands), might

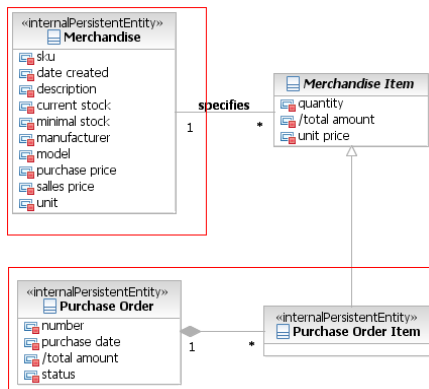


Fig. 4. Mapping between Classes and Data Functions

be manually included in the DET counting. Finally, the specialist invokes a command to calculate and store the counting summary for this Data Function. In our example, inherited attributes from the abstract Class Merchandise Item were counted as DETs.

Using OCL and Java, we created several UML Constraints associated with our stereotypes. When model validation is executed in RSA, these Constraints are checked and their violation generates error messages. Examples of the use of Constraints are: validation of the sum of selected DETs and informed DETs against the stored number of DETs, when calculating complexity; checking whether required descriptions are filled; forbidding classes from participation in two different Data Functions; consistency between FP function type and corresponding stereotype.

Counting Transactional Functions

To support Transactional Functions counting, a similar strategy was used. The FP counting specialist maps the use case event flows, modeled as activities, to Transactional Functions. Often, this mapping is one-to-one, but this depends on modeling choices, and, therefore, it is not easily amenable to full automation.

The abstract stereotype «eventFlow» was created to represent all types of event flows.

To select the list of Data Functions considered as FTRs in a Transactional Function, the RSA plug-in offers the specialist the list of all classes that are defined as Data Functions, and whose instances are roles in the collaboration that realizes the use case. For instance, if the specialist defines the main flow of the use case Manage Suppliers as an External Inquiry (EQ), the plug-in will list Purchase Order and Merchandise as candidates to FTR.

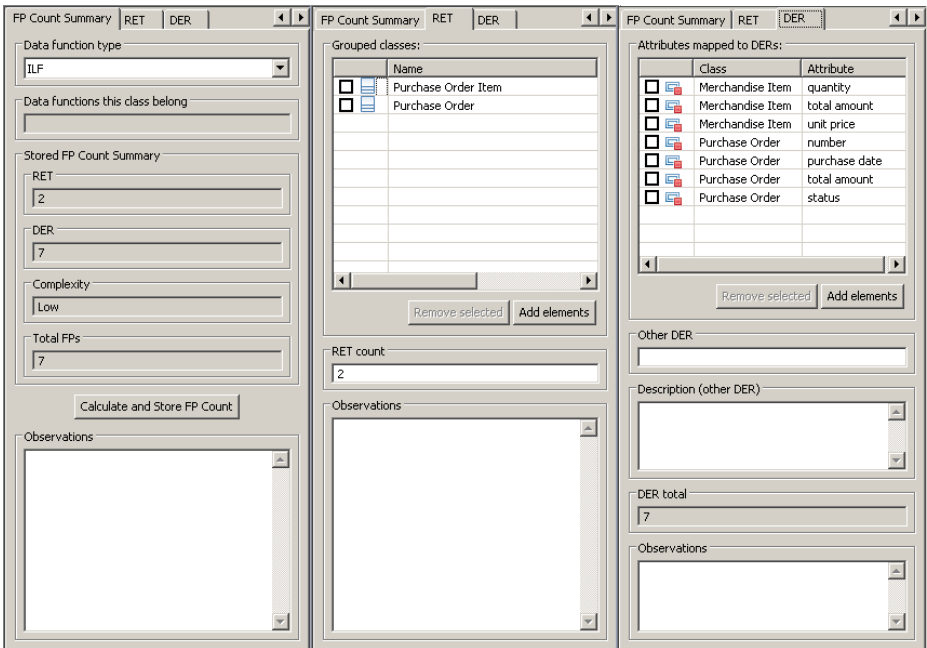


Fig. 5. RSA plug-in View to support the FP counting procedure

In order to count DETs, the RSA plug-in shows the list of all attributes from boundary classes (representing user interfaces) whose instances are roles in the collaboration that realizes the use case. Here, as in the DET counting for Data Functions, additional DETs could be manually added to the selected DETs.

For Transactional Functions, OCL constraints are also used for validation. Our tool also provides the ability to copy data from one Transactional Function to another, since many Transactional Functions share the same FTR and DET information; for instance, record insert and update, which are both EE type functions.

4.3 Assigning Constraints to Modeling Elements

In the professional Praxis version, constraints bound to stereotypes provide a quick way to enforce some rules mandated by process standards. These are metamodel-constraints, unlike stereotyped model constraints such as «businessRule» or «non-FunctionalReq», used for requirements modeling. Such metamodel constraints were coded in OCL. Evaluation is carried through the RSA. After constraint-driven validation, users can check the list of errors presented in the problem view of the tool, where they may navigate to the source of the problem and quickly fix it.

Currently, our professional Praxis version has 218 constraints distributed over 86 stereotypes. Some are:

1. Verify that the documentation fields of the elements are filled.
2. Verify that each stereotyped model element is presented in at least one diagram (this class of constraints is coded in Java, since OCL does not support diagrams and its elements).
3. Verify that mandatory stereotypes attributes are filled.
4. Check some process rules; for instance, verify whether each use case is associated with a Collaboration element through a Realization relationship.
5. Check conditional logic in filling stereotype attributes. For example, when a «field» type is set to String, the “Max Length” attribute must be filled.

Since many rules are enforced by constraints, review and inspection checklists are reduced, focusing on problems that cannot be found without human intervention.

5 Assessment

In this section we present some benefits of the usage of stereotypes and the RSA plug-in in the professional Praxis version, at Synergia. We collected benefit data for automated prototyping and for counting Function Points. All current projects at Synergia use stereotype constraints, but we were not able to compare them with older projects, since they also differ in many other aspects.

5.1 Automatic Prototype Generation

The automated prototype generation feature of the RSA plug-in was first used in a small requirements specification project (813 FP). We compared the effort needed to elaborate the specification of two current and similar projects (they used the same version of the process and the same set of tools). Table 1 presents the results. The total effort measures the work in activities that directly produce the requirements specification. Activities like management and training were excluded, but reviews and rework efforts were included.

Although there is a significant difference between these two projects in size, the data seem to indicate that automated prototype generation actually increases productivity and reduces rework. Requirements engineers that participated in both projects confirmed this impression.

Table 1. Productivity Comparison between Manual and Automated Prototypes

<i>Project</i>	<i># Use Cases</i>	<i># FP</i>	<i>Total Effort (h)</i>	<i>Productivity (PF/h)</i>
Manual Prototypes	128	2586	6727.32	0.38
Automated Prototypes	37	826	1441.83	0.57

5.2 Counting Function Points

The first project that adopted the RSA plug-in to count FP, referred here as New Project, had a size of 2586 FP distributed into 128 use cases. When the RSA plug-in became stable, we started measuring the effort for counting FP. We recorded 31 use cases data, summing up 882 FP of the 2586 FP (see Table 2).

In the New Project, each Requirement engineer was responsible for counting each use case that he/she specified, during the use case modeling. In this way, it was possible to monitor the product size in 'real time', during elicitation, and negotiate changes with the client if needed, since customers often have a predefined scope size target. Each use case FP size was verified during the requirement reviews, by other Requirement engineer and, after finishing the specification, an IFPUG certified specialist reviewed all the counting again.

The results collected from the New Project are summarized and compared to another project, referred here as Old Project. Both projects had similar characteristics (Web-based, Team maturity and size, etc.), and adopted the IFPUG counting procedures. Unlike development projects, we expected that specification projects would show an economy of scale, since smaller projects tend to have a larger overhead in activities such as start-up and closure. Nevertheless, we observed a substantial productivity increase (107%) for the counting procedure in the New Project, despite its smaller size.

During the review process, a certified specialist reviewed the estimation and detected a small deviation error (less than 2% in total FP count). This reduced error was attributed to the automated verification carried out by the stereotype constraints.

Table 2. Productivity Comparison between FP Count with and without RSA Plug-In

<i>Project</i>	<i># Use Cases</i>	<i># FP Counted</i>	<i>Total Effort (h)</i>	<i>Productivity (FP Counted/h)</i>
Old	138	4673	258.23	18.10
New	31	882	23.44	37.63

In addition to the quantitative results presented above, we also identified the following qualitative benefits:

- The RSA plug-in helps to keep up-to-date the product functional size, as measured by the FP count. As discussed before, this is a major managerial concern for Synergia. Counting FP directly in UML Model helps to keep under control the requirements changes, which may be numerous, depending on the customer needs, and to charge fair prices for them.
- It makes possible to monitor project size while its requirements are still being elicited, giving the project stakeholders a solid base to negotiate project scope.

6 Related Work

This section presents some related works about stereotype usage. In addition, we evaluated a number of related works that deployed the same functionalities to support the RE process. Our approach differs from them mainly by the use UML stereotypes. In addition, in our approach all the functionalities provided were integrated into one tool, IBM RSA, using the Problem model.

6.1 Stereotype Usage on Software Development

The idea of stereotyping was first introduced by Wirfs-Brock [7], who used this concept to classify objects according to their responsibilities as well as their modus operandi within a system. In this context, using stereotypes may be understood as a way of “branding” objects with additional intent-focused roles or properties [8]. In UML, stereotypes are used as secondary classification ways, adding properties to the elements, and as means for language extension [9] [10]. They are known to be suitable as “lightweight” extension mechanisms [10]. Instead of dealing directly with the UML metamodel, modelers can tailor the language to their own needs. With a growing number of UML tools that provide support for this extension mechanism, stereotypes begin to support more precise and detailed code generation and model transformation rules and mechanisms [11]. In addition, they may play an important role in clarifying UML diagrams [12] [13]. To summarize, stereotypes allow modelers to clarify or extend the meaning of model elements, and to introduce new modeling elements based on the elements already available in the notation.

6.2 Automatic Prototype Generation

A number of researches have been suggested for generating user interfaces from specifications of the application domain. Usually, entity-relationship models serve as input for the selection of interaction objects according to rules based on style guidelines such as Common User Access [14]. Example of researches includes Genius [15] and Janus [16].

In Genius [15], the application domain is represented by an a data model composed of views, where each view is a subset of entities, relationships, and attributes of the overall data model. A Petri-net based dialogue description specifies how these views are interconnected. From these specifications, Genius generates the user interface.

Janus [16] creates different windows of a user interface from object models. Non-abstract classes are transformed into windows, whereas attributes and methods that are marked as irrelevant for the user interface are ignored in the transformation process. Janus does not address the dynamic aspect of user interfaces.

Another approaches use scenarios as an input for the user interface prototype generation [17] [18]. Scenarios are represented in natural language [17] or in the form of collaboration diagrams [18] as defined by the Unified Modeling Language (UML). In these approaches, an intermediary representation or additional information is needed to the prototype generation, such as a graph or user interface information.

Our approach focused on the prototype generation from the UML class models. It does not require any model transformation or inclusion of additional descriptions (using non-UML notations) in the models to allow this process. This translation is carried out directly from the results produced by the analysts, which can be consumed by all the stakeholders, including our clients.

Also, our RE process uses low-fidelity prototypes, primarily as an aid to requirements elicitation, for the benefit of users who are usually not familiar with UML models. Going from low-fidelity, problem-level prototypes to high-fidelity, solution-level prototypes is a highly creative process, where usability is paramount. In this, our approach differs from several others, because it does not try to derive final user interfaces directly from the requirements, in an automated way.

6.3 Counting Function Points

In this work we defined a tool to help FP experts during FP counting based on OO requirement specifications. A fully automatic tool does not seem to be feasible, mainly because FP counting requires some human judgment [19]. Coherently with the FP concepts, we focus on the users' view of the system, and in estimating size early in the software development life cycle.

Some tools apply FPA to object-oriented (OO) models [19] [20] [21] [22]. However, a large amount of these tools deal with design models rather than requirements models [19]. Our approach derives FP directly from a UML-based requirement model, conforming to the philosophy that function points must measure the problem size, not the solution size.

Similarly to our explicit mapping from requirement models, other works also propose rules for mapping OO models to FPA models. Harpur et al. [19] propose a semi-automatic model transformation from OO requirements models to FPA models, based on heuristic rules to be applied by an FPA expert. Harpur's work presents some rules for mapping classes to Data Functions. This same mapping could be done in our work, but it involves the judgment of the FP analysis expert. To count Transactional Functions, our work relies on use case flows, rather from sequence diagrams messages. Also, since we use Unadjusted Function Points (UFP), both because they are better standardized and because they are required by the COCOMO [23] estimation tool, we do not map non-functional requirements to general system characteristics, as Harpur et al. do.

Uemura et al. [20] describe a system for automatically counting FP from a requirement/design specification based on UML diagrams. Some rules are used to extract the information needed to count Data Functions, whose complexity is based on the attributes of the corresponding selected class. For Transactional Functions, the rules look for patterns in user-system interactions, analyzing their sequence diagram messages.

Caldiera et al. [21] propose a tool which uses an object-oriented kind function points, called Object Oriented Function Points (OOPF). Our work focuses on the traditional definition of FP, since only this one is widely accepted by our customers.

Cantone et al. [22] propose conversion rules to apply FP analysis to UML, supported by an IBM Rational Rose tools. The emphasis is on UML sequence diagrams and rules applied to high-level design documents, quite from our work, which uses requirements specified at pure problem level, as officially required by FP practices.

6.4 Assigning Constraints to Modeling Elements

Much of research has been done using semi-formal specifications, especially UML diagrams. Examples of tools that check requirements consistency using semi-formal specifications are BVUML [24], CDET [25] and VIEWINTEGRA [26]. They verify the consistency between user requirements specifications and class diagrams, or between such specifications and sequence diagrams.

Other research focuses on a formal analysis technique to check the requirement specification, which implies the application of more complex concepts and representations [27] [28] [29]. For example, Hausmann et al. [28] have used a graph transformation method to detect conflicting functional requirements among use cases.

Our approach defines constraints bound to stereotypes which provide a quick way to enforce very simple rules mandated by process standards. Since the constraints to be checked are not complex, a simple OCL implementation was required without any further formalism.

7 Conclusion and Future Work

In this paper we presented our experience in applying and improving an educational software development process, Praxis, in a professional environment. We show how

the usage of UML extension mechanisms can improve software development process. As shown, stereotypes may be used not only to enhance model clarity, but in many other applications. We employed many stereotypes with tagged values to store additional information collected during modeling. In previous versions of the Praxis process, this information was usually stored in textual and spreadsheet attachments. Recording this data in a more structured way allowed us to automatically generate navigable prototypes in HTML, enhancing productivity and understanding. Stereotypes also helped to keep up-to-date the product functional size, expressed in function points, since it is directly recorded in Problem model through stereotype attributes. Data collected from recent projects have shown significant productivity improvements in the FP counting process. We also used stereotype constraints to provide automated model verification, reducing review checklists and allow model reviewers to focus on problems that need human judgment.

Regarding the limitations of this study, one could argue that our measured data are not extensive. Although this is a valid criticism, we collected feedback from the requirements engineers who were very satisfied with the improvement made in the requirement process. They also identified positive points like an expressive reduction of manual work and detected process standards' defects. Furthermore, the development was carried out for one specific proprietary tool, IBM RSA. Since we used strict UML 2 concepts, without tool-specific notation extensions, the environment should be portable to other tools that support UML 2.

As future work, we plan to invest in model transformations to generate Java code and to improve automated prototyping. For example, allowing fields to be editable based on the attributes' visibility property associated with user interfaces' state machines. We will also collect more data from our projects to observe whether quantitative benefits can be sustained.

References

1. Hall, T., Beecham, S., Rainer, A.: Requirements problems in twelve software companies: an empirical analysis. *IEE Proceedings - Software* 149(5), 153 (2002)
2. Niazi, M.: An Instrument for Measuring the Maturity of Requirements Engineering Process. In: Bomarius, F., Komi-Sirviö, S. (eds.) *PROFES 2005*. LNCS, vol. 3547, pp. 574–585. Springer, Heidelberg (2005)
3. Baziuk, W.: BNR/NORTEL: path to improve product quality, reliability and customer satisfaction. In: *Proceedings of Sixth International Symposium on Software Reliability Engineering*, pp. 256–262 (1995)
4. Pimentel, B., Filho, W.P.P., Pádua, C., Machado, F.T.: Synergia: a software engineering laboratory to bridge the gap between university and industry. In: *SSEE 2006: Proceedings of the 2006 International Workshop on Summit on Software Engineering Education*, pp. 21–24 (2006)
5. Filho, W.P.P.: Quality gates in use-case driven development. In: *International Conference on Software Engineering* (2006)
6. Pádua, W.: Using Model-Driven Development in Time-Constrained Course Projects. In: *20th Conference on Software Engineering Education & Training, CSEET 2007*, pp. 133–140 (2007)

7. Wirfs-Brock, R.: Adding to Your Conceptual Toolkit: What's Important About Responsibility-Driven Design. Report on Object Analysis and Design 1 (1994)
8. Staron, M., Kuzniarz, L., Thurn, C.: An empirical assessment of using stereotypes to improve reading techniques in software inspections. In: International Conference on Software Engineering, vol. 30(4) (2005)
9. Staron, M., Kuzniarz, L.: Properties of Stereotypes from the Perspective of Their Role in Designs. In: Briand, L., Williams, C. (eds.) MoDELS 2005. LNCS, vol. 3713, pp. 201–216. Springer, Heidelberg (2005)
10. Staron, M., Kuzniarz, L., Wallin, L.: Case study on a process of industrial MDA realization: determinants of effectiveness. *Nordic Journal of Computing* 11(3), 254–278 (2004)
11. Staron, M., Kuzniarz, L., Wohlin, C.: Empirical assessment of using stereotypes to improve comprehension of UML models: A set of experiments. *Journal of Systems and Software* 79(5), 727–742 (2006)
12. Ricca, F., Di Penta, M., Torchiano, M., Tonella, P., Ceccato, M.: The Role of Experience and Ability in Comprehension Tasks Supported by UML Stereotypes, pp. 375–384. IEEE (2007)
13. Genero, M., Cruz-Lemus, J.A., Caivano, D., Abrahão, S., Insfran, E., Carsí, J.A.: Assessing the Influence of Stereotypes on the Comprehension of UML Sequence Diagrams: A Controlled Experiment. In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., Völter, M. (eds.) MODELS 2008. LNCS, vol. 5301, pp. 280–294. Springer, Heidelberg (2008)
14. IBM, Systems Application Architecture: Common User Access – Guide to User Interface Design – Advanced Interface Design Reference (1991)
15. Janssen, C., Weisbecker, A., Ziegler, J.: Generating user interfaces from data models and dialogue net specifications. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI 1993, pp. 418–423 (1993)
16. Balzert, H.: From OOA to GUIs: The janus system. *JOOP* 8(9), 43–47 (1996)
17. Elkoutbi, M., Khriiss, I., Keller, R.K.: Automated Prototyping of User Interfaces Based on UML Scenarios. *Automated Software Engineering* 13(1), 5–40 (2006)
18. Shirogane, J., Fukazawa, Y.: GUI prototype generation by merging use cases. In: Proceedings of the 7th International Conference on Intelligent User Interfaces - IUI 2002, p. 222 (2002)
19. Harput, V., Kaindl, H., Kramer, S.: Extending function point analysis to object-oriented requirements specifications. In: 11th IEEE International Symposium on Software Metrics, pp. 10–39 (2005)
20. Uemura, T., Kusumoto, S., Inoue, K.: Function point measurement tool for UML design specification. In: Proceedings of Sixth International Software Metrics Symposium, pp. 62–69 (1999)
21. Caldiera, G., Antoniol, G., Fiutem, R., Lokan, C.: Definition and Experimental Evaluation of Function Points for Object Oriented Systems. In: Procs. IEEE METRICS 1988 (1998)
22. Cantone, G., Pace, D., Calavaro, G.: Applying function point to unified modeling language: Conversion model and pilot study. In: Proceedings of the 10th International Symposium on Software Metrics (METRICS 2004), pp. 280–291 (2004)
23. Boehm, B.W., et al.: *Software Cost Estimation with COCOMO II*. Prentice Hall PTR (2000)
24. Litvak, B., Tyszberowicz, S., Yehudai, A.: Behavioral consistency validation of UML diagrams. In: Proceedings of First International Conference on Software Engineering and Formal Methods, pp. 118–125 (2003)

25. Scheffczyk, J., Borghoff, U.M., Birk, A., Siedersleben, J.: Pragmatic consistency management in industrial requirements specifications. In: Third IEEE International Conference on Software Engineering and Formal Methods (SEFM 2005), pp. 272–281 (2005)
26. Egyed, A.: Scalable consistency checking between diagrams - the VIEWINTEGRA approach. In: Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001), pp. 387–390 (2001)
27. Heitmeyer, C.L., Jeffords, R.D., Labaw, B.G.: Automated consistency checking of requirements specifications. *ACM Transactions on Software Engineering and Methodology* 5(3), 231–261 (1996)
28. Hausmann, J.H., Heckel, R., Taentzer, G.: Detection of conflicting functional requirements in a use case-driven approach: A static analysis technique based on graph transformation. In: Proceedings of the 24th International Conference on Software Engineering ICSE 2002, pp. 105–115 (2009)
29. De Sousa, T.C., Almeida, J.R., Viana, S., Pavón, J.: Automatic analysis of requirements consistency with the B method. *ACM SIGSOFT Software Engineering Notes* 35(2), 1 (2010)
30. Batista, V.A., Peixoto, D.C.C., Borges, E.P., Pádua, W., Resende, R.F., Pádua, C.I.P.S.: ReMoFP: A Tool for Counting Function Points from UML Requirement Models. *Advances in Software Engineering* 2011 article ID 495232, 7 (2011), doi:10.1155/2011/495232

Identifying Business Rules to Legacy Systems Reengineering Based on BPM and SOA

Gleison S. do Nascimento, Cirano Iochpe, Lucinéia Thom,
André C. Kalsing, and Álvaro Moreira

Departamento de Informática, Universidade Federal do Rio Grande do Sul, 15064,
91501-970, Porto Alegre, RS, Brazil

{gsnascimento,ciochpe,lucineia,afmoreira,ackalsing}@inf.ufrgs.br

Abstract. Legacy systems include information and procedures which are fundamental to the organization. However, to maintain a legacy system is a complex and expensive task. Currently, researchers propose the legacy systems reengineering using BPM and SOA. The benefits of the reengineering using BPM and SOA are software reuse and documentation of the business processes. However, practical experiences demonstrate that reengineering using BPM and SOA are not easy to apply, because there are no tools that help the developers understand the legacy system behavior. This paper presents a tool to extract the legacy system behavior. Based on the business rules concept, we propose a tool to identify the business rules contained in legacy source code. In addition, our technique also enables the discovery of the partial order execution of the business rules during the runtime legacy system.

Keywords: business rules, legacy systems, reengineering, BPM, SOA.

1 Introduction

Legacy systems are information systems which execute useful tasks for an organization, but were developed with technologies no longer in use [1]. Legacy systems include information and procedures which are fundamental for the operation of the organization. However, to maintain a legacy system running is a complex and very expensive task. Reason is that the corresponding source code can be obsolete, hard to understand, and poorly documented.

To reduce these problems, many organizations are opting for re-implementing their legacy systems using contemporary technologies. In recent years some approaches propose the modernization of legacy systems through Business Process Management (BPM) and Service Oriented Architecture (SOA) [2], [3], [4].

These approaches use BPM and SOA in order to enable the re-implementing of legacy systems. Basically, they aim at: 1) identifying business processes which are embedded in legacy systems; and 2) implementing those business processes using BPM and calls to web services instead of completely re-implementing the legacy code

in another programming language. The major contributions these approaches can be summarized as follows:

1. Fragments of the legacy source code can be reused through of web services that are invoked of the business processes;
2. Once migrated to web services, important procedures inside the legacy source code can be reused by other organization processes;
3. An explicit documentation of the businesses processes of the organization that are modeling in a business process management system;
4. The information system execution can be both monitored and coordinated through a business process management system.

Although the benefits of software reuse and documentation of the business processes of the organization, reengineering techniques based on BPM and SOA should also be easy to apply [2], [3]. Some practical experiences demonstrate that the major problem with these techniques is in the identification of the runtime behavior of the legacy system [5], [6].

Generally, developers made this work manually, interpreting algorithms implemented in the source code and converting these algorithms to *web-services* or control-flows designed in the business process model. Thus, reengineering techniques based on BPM and SOA are dependent of the knowledge level of the developers that analysis the legacy source code. If the knowledge about the legacy system is high, the reengineering is rapid, correct and secure. Otherwise, the reengineering is slower, more expensive for the organization and more vulnerable to errors.

1.1 Legacy System Behavior

A legacy system contains a large amount of organizational knowledge [7], [8]. This organizational knowledge is specified in the source code as business rules [7], [8], [9], [10], [11]. Business rules have been defined by Bell, as abstract statements that define how the business is done or constrains some aspect of the business [9].

Usually the business rules are used to specify an information system. For example, the statement – *“If the purchase amount is greater than US\$ 3.000, then the system must calculate a discount of 10%”* – is a business rule that defines an action of the information system. During the implementation phase of the system, the developers use these statements to write the source code of the information system.

Therefore, we can say that the behavior of an information system is determined by business rules that are implemented in the source code [7], [8], [10]. Thus, to identify the behavior of a legacy system is necessary to discover the business rules implemented in its source code.

1.2 Proposed Solution to Identification of the Legacy System Behavior

Based on the business rules concepts, we define a semi-automatic technique to identify the legacy system behavior. This technique allows the semi-automatic identification of the business rules implemented in the legacy source code and

automatic discovery of the partial order execution of these business rules during the runtime legacy system. The identification of the business rules is semi-automatic, because it requires human intervention to complete the identification of the rules.

The proposed technique consists of five steps:

- **Step 1:** Identify the business rules in the legacy source code, annotating the starting and ending points of their encoding as shown in Fig. 1(a). The annotations are two lines of code introduced in the source code. These lines generate new entries in the log file during the runtime legacy system.
- **Step 2:** Run the legacy system to generate a log file that contains the entries of start and end of the business rule annotated in the source code. Legacy system must be executed several times, with different data sets. Fig. 1(b) shows an example of log file generated during the runtime legacy system.
- **Step 3:** Mine the log file, with a log mining algorithm, which obtains the partial order execution of the business rules. Fig. 1(c) shows the partial order that can be expressed in a business process model.

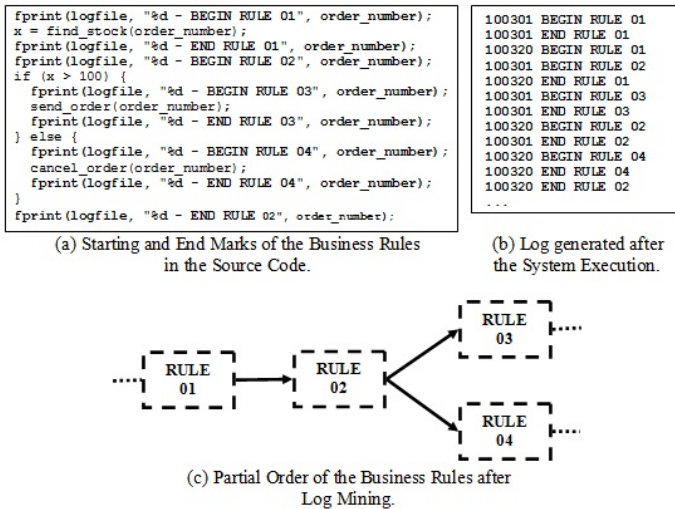


Fig. 1. Example of the steps to discovery the partial order execution of the business rules implemented in a legacy system

- **Step 4:** Validate the partial order execution of the business rules with expert users of the legacy system.
- **Step 5:** Map the partial order execution of the business rules to a business process model. A business process also can be specified and implemented with business rules [13], [14]. Thus, we can use the business rules as invariants to map parts of the legacy source code into business process fragments. Therefore, the model obtained through the log mining does not represent a complete business process model. It is need to replace the business rules identifiers in the Fig. 1(c) by business process fragments that implement these business rules in a business process model.

This paper presents to Step 01 of the proposed technique. The remaining steps have been presented in previous papers [4], [15].

We studied the business rules concept in the context of legacy systems. Based on the literature, we define a business rules ontology that groups the main business rules types used in legacy systems. After the definition of the ontology, we verify how the business rules types are implemented in the legacy source code, i.e., those programming language statements are commonly used to implement each business rule type. With this information, we create a tool that analyses the legacy source code, identifying the business rules implemented in this source code. Altogether, the main contributions of this paper are:

- Creation of an business rules ontology with main business rules types that are used in legacy systems;
- Definition of a tool to identify the business rules implemented in the legacy source code;
- Presentation of a technique to map the legacy system behavior into business process models.

The remainder of this paper is organized as follows: Section 2 presents the related works. Section 3 defines the business rules concept in the context of legacy systems. Section 4 defines the ontology that is used in the paper. Section 5 defines the programming language statements that are used to implement the business rules type. Section 6 demonstrates the soundness and correctness of the ontology. Section 7 presents the tool proposed to identify the business rules implemented in a legacy system. We conclude with a summary and outlook in Section 8.

2 Related Work

Business rules identification from legacy systems is a common challenge that has been addressed in the literature for many years. Business rules identification is main challenge from many approaches for the modernization of legacy systems.

To identify and extract the business logics inside the source code, [8] and [10] defined business rules as functions with conditions, and consider 1) output and variable, 2) data flow and dependency, and 3) program slicing to reduce the searching scope in the source code. In [8], code restructuring and use case identification were used to extract business logics from COBOL programs. [10] proposed a manual approach to extract the business logics from the source code and showed the effectiveness of human identification.

These related works allow identifying business rules from legacy but don't allow the identification of business processes associated to these rules. In our research, we pretend to identify the business processes implemented implicitly in the legacy source code. Thus, we use the business rules as invariants to map source code fragments into business activities of the business processes.

Recent researches propose the use of business rules to model business processes. In [14] the business rules are used to specify conditional flows in web service compositions. In [17] the business activities are process elements and construct flows

based on ECA business rules. By chaining these rules a flow is constructed that can be translated to an executable process specification. Also, in [13] business rules are related to workflow patterns with the use of ECA rules.

Therefore, in this paper, we present a technique to identify different business rules types. Each business rule type identified represents determined business semantics of the organization. We map the business rules of the legacy code into business activities from business processes.

Regarding the other researches in business rules identification, we identify more business rules types. In [8] and [10] are identified conditional business rules that determine the control flow of the legacy system. However, to model a business process are required other business activities, such as user interaction activities and automatic activities. In our approach we identify the conditional business rules and others business rules types, which can later be mapped to specific business activities to model the business processes implemented in the legacy system.

3 Business Rules

Typically, business rules are used to specify an information system. The business rules are statements that guide developers in the encoding of the information system. Therefore, we can say that the behavior of an information system is determined by business rules that are implemented in the source code [7], [8], [10].

Fig. 2 shows the declaration of a business rule that specifies the behavior of an information system (a). During the encoding of the system, the developers write this business rule in the source code using programming language statements (b).

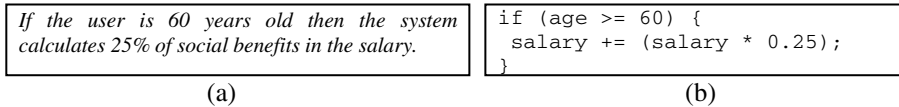


Fig. 2. (a) Textual Description of a Business Rule and (b) Implementation of the Business Rule using the C Programming Language

Business rules can be classified in types [7], [11]. A business rule type defines a set of words or expressions that must be used to write a business rule in natural language [11]. These words or expressions are translated into programming language statements during the encoding of the information system.

Therefore, to identify business rules in the source code is necessary to know the business rules types, what the properties of each business rule type and how these properties are implemented using an imperative programming language.

Thus, we define a business rules ontology that has the main business rules types found in legacy systems. The ontology was defined with different business rules classification schemes found in the literature [7], [11], [12], [16], [18].

The business rules types of the classification schemes were searched in the technical documentation (e.g.: textual specification) of legacy systems found in the Brazilian Public Software Portal [19]. In the documentations we search textual

sentences that describe business rules (Fig. 3(a)). After, these business rules were classified in the business rules classification schemes found in the literature. Thus, we identify the main business rules types used in the legacy systems studied in this work.

For example, the textual sentence “*To request a social benefit the user **must** be older than 60 years*” was found in the documentation of a legacy system. This sentence has an obligation expression (“**must**”) and a conditional expression (“**older than 60 years**”), which are features of a constraint business rule defined in the classification scheme of Ross [18]. Therefore, this textual sentence was classified as a constraint business rule of the classification scheme of Ross [18].

After the classification of the business rules contained in the documentation of the legacy systems, we create ontology with the main business rules types found in these systems (Fig. 3(b)). Subsequently, the ontology is the knowledge base to a tool of identification of the business rules implemented in the source code of legacy systems.

The classes of the ontology represent the business rules types and the attributes represent expressions of the natural language that are used to write a business rule of a particular business rule type. Fig. 3(b) shows an example of class of our ontology.

With the definition of the ontology, we verify how the business rules found in the documentation have been implemented in the source code of the legacy systems. We observe which statements of an imperative programming language were used to implement the business rules types (Fig. 3(c)).

Thus, we define an equivalent class to each business rule class of the ontology. The equivalent class represents the implementation of a particular business rule type. In the equivalent class the attributes are mapped to programming language statements, as shows the Fig. 3(d). The programming language statements normally are used to implement a particular business rule type.

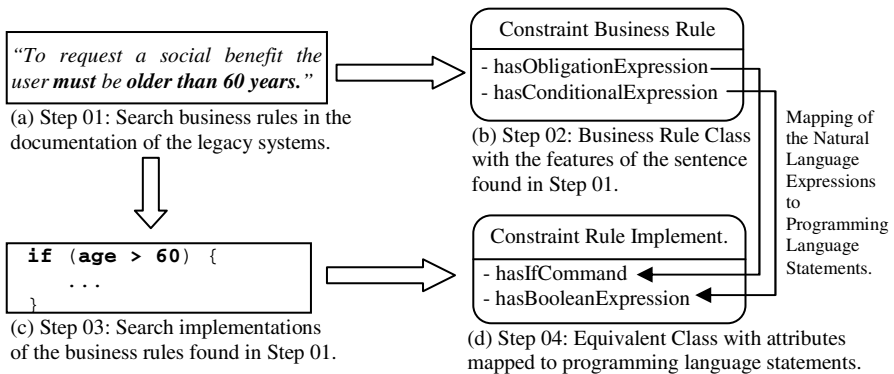


Fig. 3. Steps followed to define the business rules ontology and the equivalent classes

4 Business Rules Ontology

To find the business rules types that are used in legacy systems, we use the technical documentation of some systems available in the Brazilian Public Software Portal [19]. As shown in Fig. 3(a), we search in the documentations by textual sentences that

specify the business rules of such systems. We found 417 sentences that describe business rules of these systems.

These sentences were categorized into the business rules types found in the classification schemes defined in the literature [7], [11], [12], [16], [18]. During the categorization we observe that the classification scheme defined by Ross [18] is the most complete. All textual sentences found in the documentations of the systems were classified into some business rule type defined by Ross [18]. Therefore, this classification scheme was chosen to define our business rules ontology.

Section 5 presents details of the results obtained in the mining of the legacy systems of the Brazilian Public Software Portal [19]. This section presents two statistics: support and confidence. Support verifies the probability of a business rule type appears in the legacy systems. While the confidence checks the probability of a business rule type to be implemented in the legacy source code, according with the programming language statements defined in the equivalent classes (Fig. 3(d)).

From the classification scheme defined by Ross [18] we generate ontology to business rules specified in natural language. For each business rule type were created a class in the ontology, where the attributes represent expressions that are commonly used to write a business rule of this type in natural language.

According to the classification scheme defined by Ross [18], a business rule specifies a relationship among business concepts.

A business concept is a term that expresses an idea or notion about the business of an organization. A business concept can represent an object or an entity in the business. It is an abstract representation of the reality of a particular business [18].

Business concepts are related through expressions that indicate actions (e.g. verbs). In the example “*The customer buys cars*” there are two business concepts, “*customer*” and “*cars*”, that are related through the action expression, “*buys*”. Thus, we can say that a business rule must have at least two business concepts and an action expression to relate the business concepts.

Thus, the ontology starts with the “*Business Rule*” and “*Business Concept*” classes as presented in Fig. 4. Business Rule Class defines a generic business rule, which must have two or more business concepts and one action expression. The “*hasBusinessConcept*” attribute indicates the requirement of the business concepts to a sentence textual to be a business rule. The “*concepts*” attribute indicates the business concepts involved in the textual sentence. While the “*hasActionExpression*” attribute indicates the need of an action expression relating the business concepts.

According to Ross [18], the business rules are divided into six classes: Constraint Rule; Computation Rule; Derivation Rule; Enabler Rule; Copier Rule; and Executive Rule. Fig. 4 shows our ontology, where the classes defined by Ross [18] are specializations of the “*Business Rule*” class.

The features of the business rules types defined by Ross [18] are transformed into attributes of the classes in Fig. 4. For example, the “*Constraint Business Rule*” class has the “*hasObligationExpression*” attribute indicating the requirement of an obligation expression in textual sentences that specify constraint business rules.

Note in Fig. 4 that the “*Computation Business Rule*” class was specialized in “*Persistence Business Rule*” and “*Mathematical Business Rule*”, because this

business rule type is characterized by a mathematical or persistence action. Thus, we specialize the “*Computation Business Rule*” class to eliminate this ambiguity. If the textual sentence has a mathematical action (e.g.: to sum) will be classified as “*Mathematical Business Rule*”. Otherwise, the textual sentence has a persistence action (e.g.: to save) and it will be classified as “*Persistence Business Rule*”.

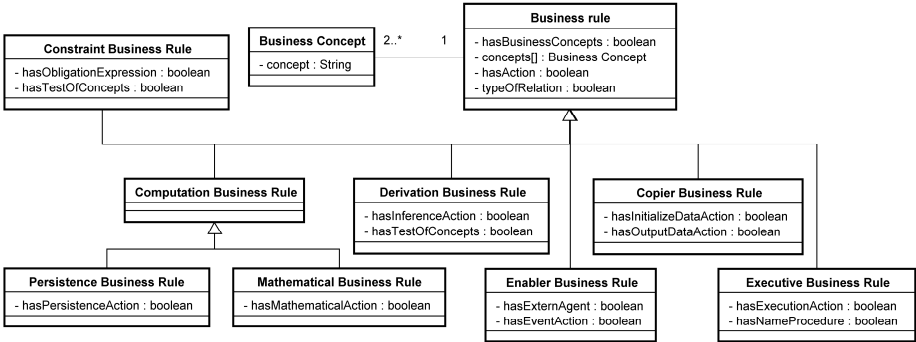


Fig. 4. Business Rules Ontology with the Features in Natural Language

5 Implementation of the Business Rules Classes

Now, for each business rule class in the ontology, we propose an equivalent class with the implementation features of such business rule class in an imperative programming language. The attributes of the equivalent class refers to programming language statements that are normally used to implement a business rule of that particular class.

The equivalent classes are used in the tool that analyses the legacy source code, identifying and annotating the business rules of this system.

To define the equivalent classes, we search in the source code of the legacy systems studied in this work, the implementations of the 417 business rules found in the technical documentation of the systems (Fig. 3(c)). Thus, we observe the programming language statements used to implement the business rules classes defined in the ontology of the Fig. 4 (Fig. 3(d)).

Legacy systems studied in this work are written in C, COBOL, and PHP programming language. However, we can generalize the programming language, because the semantics of the statements used in the implementation of the business rules is identical in the three programming languages.

5.1 Programming Language

Fig. 5 shows the programming language grammar used in the paper. This grammar is an extension of the programming language defined by Plotkin [20], with the main statements found in an imperative programming language.

```

e ::= n | s | v | e1 + e2 | e1 - e2 | e1 * e2 | e1 / e2
b ::= true | false | e1 = e2 | e1 > e2 | e1 < e2 | b1 and b2 | b1 or b2
c ::= v := e | c1;c2 | call proc | v := call func | print(e) |
      read(v) | if b then c end | if b then c1 else c2 end |
      while b do c end | exec sql query | begin sql c end sql
    
```

Fig. 5. Grammar Programming Language

In the grammar is used $e, e_1, e_2 \dots$ to represent arithmetic expressions, $b, b_1, b_2 \dots$ to represent boolean expressions, as well as $c, c_1, c_2 \dots$ to represent the commands of the programming language.

In grammar also is used “ n ” for numbers, “ s ” for strings, “ v ” for variables, “ $proc$ ” and “ $func$ ” to procedures and functions. In addition, “ $query$ ” is a valid SQL statement, i.e., “ $query$ ” is a sentence *INSERT, UPDATE, DELETE* or *SELECT* of the structured query language used to access a database.

The behavior of the programming language statements presented in Fig. 5 is defined with conceptual graphs [21]. Conceptual graphs have been proposed in [21] to represent a particular meaning in a format logically precise, humanly readable and computationally interpretable. Basically, a conceptual graph consists of two nodes types: **Concepts** and **Relations**. Concepts are arranged inside of the rectangles and are connected through the relations nodes. Relations are arranged into diamonds and connect the concepts by all sides that they can be linked.

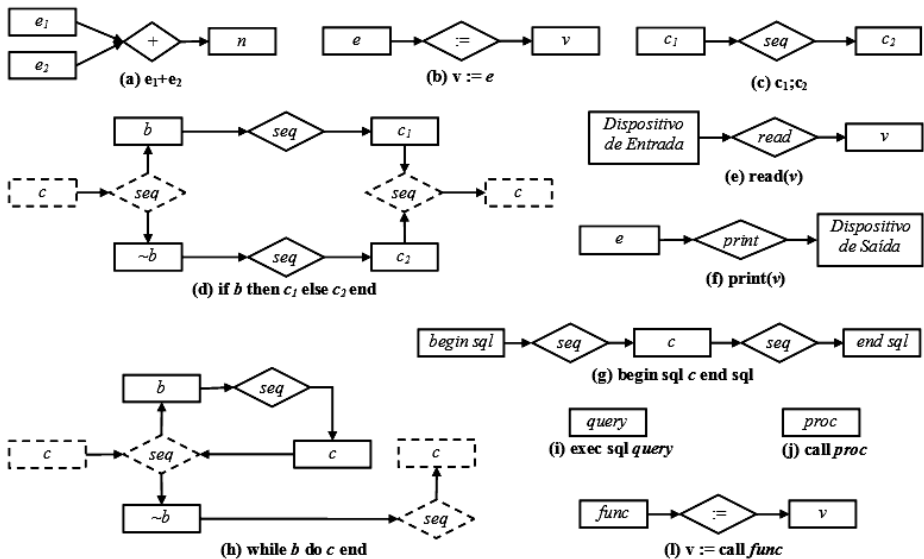


Fig. 6. Conceptual Graphs to the Programming Language Statements

For each programming language statement is defined a conceptual graph that represents your behavior (Fig. 6). Subsequently, these conceptual graphs are used to generate conceptual graphs that represent the behavior of the business rules classes.

The conceptual graphs also will be important to the tool of identification of the business rules implemented in the source code. Our tool uses the graphs defined to the programming language statements to generate a conceptual graph that represents the behavior of the legacy source code. In the conceptual graph of the legacy source code we search fragments that represent the behavior of the business rules.

In the semantics of the programming language the *boolean* expressions (“*b*”), numeric expressions (“*e*”), the numbers (“*n*”), variables (“*v*”) and commands (“*c*”) are the concepts. Thus, they are expressed by rectangles in conceptual graphs.

For example, in “ $e_1 + e_2$ ” the expressions “ e_1 ” and “ e_2 ” are concepts, while the sum operator (+) is a relation that links these concepts to generate a new concept as shows the Fig. 6(a).

Note that the concepts “ e_1 ” and “ e_2 ” can be substituted to another *subgraph* when “ e_1 ” or “ e_2 ” are other mathematical expressions. This substitution is done until the expression to be a variable (“*v*”) or number (“*n*”). The assignment command has the graph shown in Fig. 6(b). The Fig. 6 shows the others conceptual graphs.

5.2 Mapping the Business Rules Classes into Equivalent Classes

The next step is to map the business rules classes presented in Section 3 into equivalent classes. For each business rule class of the ontology is defined an equivalent class. In the equivalent class the attributes refer to programming language statements, which usually are used to implement the business rules of this class in a programming language.

For example, in the “*Business Rule*” class, the “*concepts*” attribute is transformed “*variables*” which is the component used in the implementation of the business concepts using a programming language. Usually the business concepts are implemented in the source code of an information system using the variable concept of the programming language.

The equivalent classes were based in the implementations of the 417 business rules found in the documentations of the legacy systems studied in this work. We search in the source code files of these systems for code fragments that implement the business rules found in the documentations. Thus, we can determine the mapping from natural language expressions used in the textual sentences into programming language statements, which usually are used in the implementation of each business rule class.

We also define a conceptual graph for each equivalent class. This graph represents the behavior of the business rule class. We use the graphs defined to the programming language statements to generate a conceptual graph for each equivalent class. For example, if a particular class is implemented in the programming language by an “*if*” command followed by an assignment command, then the conceptual graph of this business rule class is the union of the graphs of the Fig. 6(d) and Fig. 6(b).

The first mapping is of the “*Business Concept*” class. The business concepts are implemented in source code using variables. Thus, this class is mapped to a “*Variable*” class, with the “*identifier*” attribute equivalent to the “*name*” attribute of the “*Business Concept*” class.

The “*Business Rule*” class is mapped to the “*Business Rule Impl.*” class. This same naming is used for the others business rules classes of the Fig. 4.

The “*hasBusinessConcepts*” attribute of the “*Business Rule*” class is mapped to the “*hasVariables*” attribute in the “*Business Rule Impl.*” class. Similarly, the relationship of “*Business Rule*” class to “*Business Concept*” class is mapped to a relationship between the “*Business Rule Impl.*” and “*Variable*” classes.

A business rule is implemented through statements that manipulate variables (business concepts). Consequently, the programming language statements corresponding to action expressions contained in the “*Business Rule*” class. Therefore, the “*hasActionExpression*” attribute in “*Business Rule*” class is mapped to the “*hasStatment*” attribute in class “*Business Rule Impl.*” as shows Fig. 7.

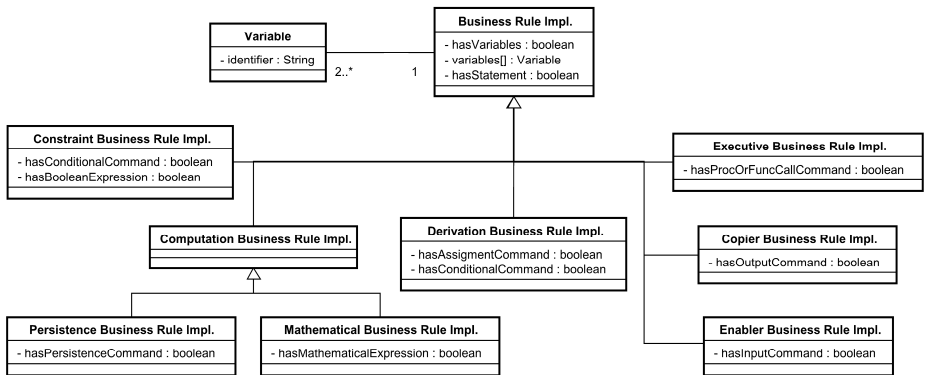


Fig. 7. Equivalent Classes with the Features of Implementation of the Business Rules

In the Fig. 7 are presented the equivalent classes for each business rule class shown in Fig. 4. These definitions follow the same logic presented to the “*Business Rule*” class, observing the programming language statements used in the implementation of the 417 business rules found in the documentation of the legacy systems used in the context of this study.

Note in Fig. 7 that the equivalent classes have attributes that indicate the programming language statements that normally are used to implement the business rules found in the legacy systems studied in this work. Thus, it is possible to generate a conceptual graph to represent the behavior of each business rule class.

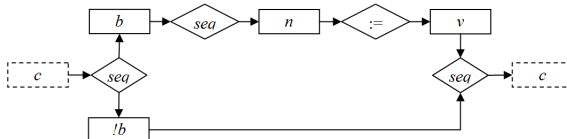


Fig. 8. Conceptual Graph to “*Derivation Business Rule Impl.*” Class

For example, in “*Derivation Business Rule Impl.*” class is characterized by one “*if*” command and one assignment command. Therefore, we use the conceptual

graphs of these statements to represent the behavior of this business rule class as shown in Fig. 8.

6 Support and Confidence

To ensure that the ontology and the equivalent classes are corrects, we collect the support and confidence statistics [22].

The support and confidence are statistics computed in the mining of association rules in records of the database [22]. They allow verifying the probability of an association rule appears in a data set.

These statistics were adapted to the context of this work. We verify the probability of the business rules classes appear in a legacy system and the probability of them being implemented in the source code using the programming language statements presented in the equivalent classes (Fig. 7).

The support and confidence were collected in the analysis of the legacy systems from the Brazilian Public Software Portal [19]. These systems are implemented in different technologies (e.g., COBOL, C and PHP), totaling about 30 thousand lines of source code and 60 pages (e.g., A4 size) of technical documentation.

The support statistic shows the probability of a business rule class to be found in the legacy systems analyzed in this work.

After identifying and classifying the business rules contained in the documentation of the legacy systems, we count the occurrences of each business rule class. We determine the support statistic to a particular business rule class, dividing the number of occurrences of the business rule class by the total number of business rules found in the legacy systems. Fig. 9(a) shows the formula used to the support statistic.

C = Business Rule Class.

TBR = Total Number of Business Rules in the Legacy Systems.

O(C) = Occurrences of a Particular Business Rule Class in the Legacy Systems.

OI(C) = Occurrences of a Particular Business Rule Class Implementation in the Source Code.

$$\text{Support (C)} = O(C) / \text{TBR}$$

(a)

$$\text{Confidence (C)} = OI(C) / O(C)$$

(b)

Fig. 9. Formulas to the Support and Confidence Statistics of the Business Rules Classes

The confidence statistic verifies if the implementation features of the business rules classes are consistent, i.e., the probability of an business rule instance to be implemented through the programming language statements defined for its equivalent class of the Fig. 7.

For example, of the total number of business rules classified as copier business rule, we verify how many occurrences have been implemented with the output command (*print(v)*). The confidence statistic is obtained dividing the number of occurrences of the copier business rule implemented with the output command by the total number of occurrences of the copier business rule found in the legacy systems. Fig. 9(b) shows the formula used to calculate the confidence statistic.

Table 1 shows the results of the support and confidence statistics for each business rule class. The results summarize the data collected for all legacy systems analyzed in the context of this work.

Note in the Table 1 that the confidence of the equivalent classes is greater than 90% (except the mathematical business rule that is 88%). This demonstrates that in the context of the legacy systems studied in this work, there are a high percentage of business rules that were implemented using the programming language statements indicated in the equivalent classes of the Fig. 7.

Table 1. Results collected to Support and Confidence Statistics

	Total Number of Business Rules: 417			
Business Rule Class (C)	O(C)	OI(C)	Support	Confidence
Constraint Business Rule	129	124	0,30	0,96
Mathematical Business Rule	9	8	0,02	0,88
Persistence Business Rule	123	120	0,29	0,97
Derivation Business Rule	32	30	0,07	0,93
Copier Business Rule	46	44	0,11	0,95
Enabler Business Rule	21	20	0,05	0,95
Executive Business Rule	57	54	0,13	0,94

Also note that the confidence of the mathematical business rule was 88% because of the low number of instances of this business rule type. This number is due to features of the legacy systems used in this work. Note that the computation business rule defined by Ross [18], in our ontology was specialized to “*Mathematical Business Rule*” and “*Persistence Business Rule*” class. Therefore, we can sum the results obtained to mathematical and persistence business rules, obtaining a confidence of 96% to the “*Computation Business Rule*” class.

The confidence was not higher, because some business rules identified in the legacy systems have features of two or more classes (i.e., ambiguous classification). However, this problem can be avoided by a human review of the business rules with ambiguous classification. The tool of identification of the business rules can indicate to an expert human the business rules with ambiguous classification. This reduces the manual work in the analyses of the source code.

All business rules classes have occurrences in the legacy systems studied in the work. Business rules that have more occurrences are the constraint and computation business rules, which together has a support of 61%.

The enabler business rule appears 5% because these rules are user inputs, which are usually grouped into electronic forms. Thus, one electronic form is considered only one enabler business rule.

7 Tool to Identify Business Rules in the Source Code

Based on the business rules ontology and in the conceptual graphs of the business rules implementation classes, we define a tool to identify and to annotate the business rules contained in the source code files from a legacy system.

The tool receive as input a source code file and return as output the source code file with the annotations from the starting and ending points of the business rules contained in the source code, as presented in the Fig. 1(a).

Fig. 10 shows the architecture of the tool divided in two modules: syntactic analysis module - used to the *parsing* of the source code; and semantic analysis module - used to identify the code fragments that implement the business rules.

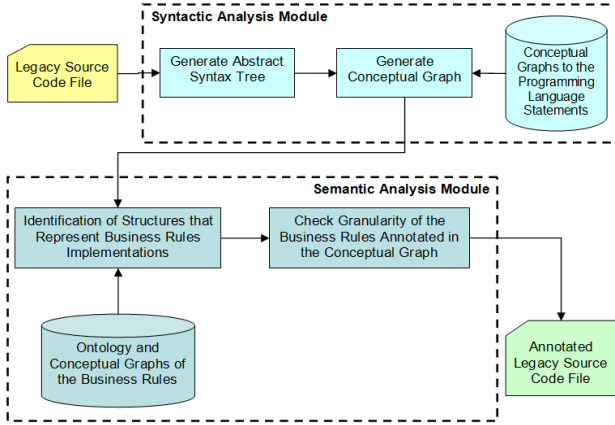


Fig. 10. Architecture of the Proposed Tool to Identify Business Rules in Legacy Source Code

The tool has four steps as shown in Fig. 10:

- **Generate Abstract Syntax Tree:** the first step is to generate an abstract syntax tree (AST) to perform the *parsing* of the legacy source code. The *parsing* generates a data structure that facilitates the interpretation of the source code, verifying the correctness of the code, i.e., if the source code has no syntax problems. In this step, we use algorithms that generate an n -ary tree, where the nodes represent the programming language statements used in the source code [23].
- **Generate Conceptual Graph of the Legacy Source Code:** the abstract syntax tree is transformed in a conceptual graph that represents the legacy source code. To create the conceptual graph of the legacy source code are used graphs defined to the programming language statements, as shows the Fig. 6. For each statement found in the abstract syntax tree, the conceptual graph of the source code receives the addition of the *subgraph* that represents this programming language statement (Fig. 6).
- **Identification of Structures that Represent Business Rules Implementations:** the next step is traverse conceptual graph annotating structures that represent business rules implementations. The identification is made through the search of *subgraphs* with identical structure to the graphs defined to business rules implementation classes. For example, to the “*Derivation Business Rule Impl.*” class our tool must traverse the conceptual graph of the source code, looking for *subgraphs* with the same structure of the

conceptual graph defined in the Fig. 8. When the tool finds a *subgraph* that represents the business rule implementation, it annotates in the conceptual graph of the source code, all the components that make this business rule.

- **Check Granularity of the Business Rules Annotated in the Conceptual Graph:** the last step the tool search in the conceptual graph of the source code, by business rules that can be united in only one business rule. For example, a sequence of calculations can be annotated in the graph as different business rules, but to the business this sequence should be only one indivisible calculation. The analysis of the granularity of the business rules is done through heuristics and algorithms that analyze the dependence of the variables involved in the business rules.

At the end of the granularity analysis the tool annotates the legacy source code file with the information annotated in the conceptual graph of the source code, generating the final result that is the annotated source code file.

8 Summary and Outlook

In this paper, we present a tool of identification of the business rules implemented in the source code of legacy systems. Our tool is based on a business rules ontology that contains the main business rules types that can be found in legacy systems.

We demonstrate statistically what business rules classes are mostly found in legacy systems. We also show that the business rules classes are normally coding using standard programming language statements.

All business rules found in the legacy systems were classified in the classes of our ontology. However, we can not say that there are only these rules classes. Thus, our ontology has a generic business rule class, where the tool classifies a source code fragment is not ranked in any business rule class. After the identification, a human expert analyzes the source code fragment and proposes changes in the ontology.

Now this tool of business rules identification will be used in the reengineering technique based on BPM and SOA, as presented in Section 1.3. The next phase of the research will be a full case study of the reengineering technique, applying the technique in an operational legacy system. Thus, we can generate statistics to demonstrate the effectiveness of the technique proposed in Section 1.3.

Acknowledgements. We are very grateful for the SticAmSud and PNPD Program from the Brazilian Coordination for the Improvement of Graduated Students (CAPES).

References

1. Ward, M.P., Bennett, K.H.: Formal methods for legacy systems. *Journal of Software Maintenance and Evolution* 7(3), 203–219 (1995)
2. Borges, M.R.S., Vincent, A.F., Carmen Penadés, M., Araujo, R.M.: Introducing Business Process into Legacy Information Systems. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) *BPM 2005*. LNCS, vol. 3649, pp. 452–457. Springer, Heidelberg (2005)

3. Kamoun, F.: A Roadmap towards the Convergence of Business Process Management and Service Oriented Architecture. *ACM Ubiquity* 8(14), 79–84 (2007)
4. Nascimento, G.S., Iochpe, C., Thom, L.H., Reichert, M.: A Method for Rewriting Legacy Systems using Business Process Management Technology. In: 11th International Conference on Enterprise Information Systems (ICEIS), pp. 57–62. ISAS, Milan (2009)
5. Acharya, M., Kulkarni, A., Kuppili, R., Mani, R., More, N., Narayanan, S., Patel, P., Schuelke, K.W., Subramanian, S.N.: SOA in the Real World – Experiences. In: Benatallah, B., Casati, F., Traverso, P. (eds.) *ICSOC 2005*. LNCS, vol. 3826, pp. 437–449. Springer, Heidelberg (2005)
6. Brahe, S.: Early Experiences on Adopting BPM and SOA: An Empirical Study. Technical Report TR-2007-96, IT University of Copenhagen (2007)
7. Kolber, A., Hay, D., Healy, K.A., et al.: Defining Business Rules - What Are They Really? Technical Report, Business Rules Group (2000)
8. Wang, X., Sun, J., Yang, X., He, Z., Maddineni, S.: Business Rules Extraction from Large Legacy Systems. In: 8th European on Software Maintenance and Reengineering (CSMR), pp. 249–257. IEEE Press, Washington, DC, USA (2004)
9. Bell, J., Brooks, D., Goldbloom, E., Sarro, R., Wood, J.: Re-Engineering Case Study – Analysis of Business Rules and Recommendations for Treatment of Rules in a Relational Database Environment. Technical Report, Information Technologies Group (1990)
10. Sneed, H.M., Erdos, K.: Extracting Business Rules from Source Code. In: 4th IEEE IWPC 1996, pp. 240–247. IEEE Press, Washington, DC, USA (1996)
11. Ross, R.G.: *The Business Rule Book: Classifying, Defining and Modeling Rules*. Business Rule Solutions Inc., Houston (1997)
12. Weiden, M., Hermans, L., Schreiber, G., van der Zee, S.: Classification and Representation of Business Rules. Technical Report, University of Amsterdam (2002)
13. Knolmayer, G.F., Endl, R., Pfahrer, M.: Modeling Processes and Workflows by Business Rules. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) *BPM 2000*. LNCS, vol. 1806, pp. 16–29. Springer, Heidelberg (2000)
14. Charfi, A., Mezini, M.: Hybrid Web Service Composition: Business Processes meet Business Rules. In: 2th International Conference on Service Oriented Computing, pp. 30–38. ACM, New York (2004)
15. Kalsing, A.C., Nascimento, G.S., Iochpe, C., Thom, L.H., Reichert, M.: An Incremental Process Mining Approach to Extract Knowledge from Legacy Systems. In: 14th International IEEE EDOC Conference, pp. 79–88. IEEE Press, Washington, USA (2010)
16. Von Halle, B.: *Business Rules Applied*. John Wiley & Sons Inc., Hoboken (2001)
17. Lee, S., Kim, T., Kang, D., Kim, K., Lee, J.: Composition of executable business process models by combining business rules and process flows. *Journal Expert System with Applications* 33, 221–229 (2007)
18. Ross, R.G.: RuleSpeakTM – Templates and Guidelines for Business Rules. *Business Rules Journal* 2(5) (2001)
19. Portal do Software Público Brasileiro, <http://www.softwarepublico.gov.br>
20. Plotkin, G.D.: A Structural Approach to Operational Semantics. Technical Report, University of Aarhus, Denmark (1977)
21. Sowa, J.F.: *Conceptual Graphs* (2004), <http://www.jfsowa.com/cg>
22. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.: Fast discovery of association rules. In: Fayyad, U.M., et al. (eds.) *Advances in Knowledge Discovery and Data Mining*, pp. 307–328. AAAI Press, Menlo Park (1996)
23. Parr, T.J., Quong, R.W.: ANTLR: A predicated-LL(*k*) parser generator. *Software: Practice and Experience* 25, 789–810 (1995)

Abstraction Analysis and Certified Flow and Context Sensitive Points-to Relation for Distributed Programs

Mohamed A. El-Zawawy

College of Computer and Information Sciences, Al-Imam M. I.-S. I. University
Riyadh, Kingdom of Saudi Arabia

and

Department of Mathematics, Faculty of Science, Cairo University
Giza 12613, Egypt
maelzawawy@cu.edu.eg

Abstract. This paper presents a new technique for pointer analysis of distributed programs executed on parallel machines with hierarchical memories. One motivation for this research is the languages whose global address space is partitioned. Examples of these languages are Fortress, X10, Titanium, Co-Array Fortran, UPC, and Chapel. These languages allow programmers to adjust threads and data layout and to write to and read from memories of other threads.

The techniques presented in this paper have the form of type systems which are simply structured. The proposed technique is shown on a language which is the *while* language enriched with basic commands for pointer manipulations and also enriched with commands vital for distributed execution of programs. An abstraction analysis that for a given statement calculates the set of function abstractions that the statement may evaluate-to is introduced in this paper. The abstraction analysis is needed in the proposed pointer analysis. The mathematical soundness of all techniques presented in this paper are discussed. The soundness is proved against a new operational semantics presented in this paper.

Our work has two advantages over related work. In our technique, each analysis result is associated with a correctness proof in the form of type derivation. The hierarchical memory model used in this paper is in line with the hierarchical character of concurrent parallel computers.

Keywords: abstraction analysis, certified code, flow and context sensitive points-to relation, distributed programs, semantics of programming languages, operational semantics.

1 Introduction

The need for distributed programs [20,21] for graphics processors and desktops was magnified by the creation of multi-core processors which therefore greatly affected the software development. Large parallel systems were built using multi-core processors. These systems have memory that uses message passing and

that is cache-coherent shared, direct-access (DMA or RDMA) distributed, and hierarchical [17,18]. An appropriate address space model for machines equipped with multi-core processors is the partitioned global model (PGAS). Examples of DPLs, distributed programming languages that program machines equipped with multi-core processors, that use PGAS are Unified Parallel C (UPC), X10, Chapel, and Titanium which is based on Java.

The objective of pointer analysis [10,8,13,9] is to calculate for every variable at each program point of the program the set of addresses to which the variable may point. Pointer analysis of DPLs is a complicated problem as these languages allow pointers to shared states. A common query concerning pointer analysis of DPLs is whether the access of a given pointer can be proven to be restricted to a specific region of the memory hierarchy. Such information is useful in many directions including the following: (a) pointer representation – fewer bits are needed to represent pointers with restricted domains; (b) improving performance of software caching systems – coherence protocols may be limited to a proper subset of processors; (c) allowing data race to ignore pointers that access private data of a thread; (d) identifying pointers that access data on chip multiprocessor and hence need ordering fences [23].

The algorithmic style is the typical manner to analyze distributed program. This manner relies on data-flow analyses and works on control-flow graphs – intermediate representations of programs. The type-systems manner is an alternative style for analyzing distributing programs and it works on the syntactical structure of programs [10,8,13,9]. This latter manner is the convenient approach to analyze distributed programs [2,16,27] when justifications for the correctness of analysis results are required to be delivered together with analysis results. These communications are required to clarify the way the analysis results were obtained. Certified code is an application that requires machine-checkable, simple, and deliverable justifications. Justifications provided by the type-systems manner have the form of type derivations which are user-friendly. The type systems approach has the advantage of relatively simple inference rules.

In previous work [10,8,13,9,11,6], we have shown that the type-systems approach is indeed a convenient framework for achieving pointer analysis of *while* languages enriched with pointer and parallel constructs. In this paper, we show that this approach extends also to the complex problem of pointer analysis of distributed programs. Many factors cause the complexity of the problem; (a) the use of shared memory in distributed programs allows pointing to these locations and (b) the interference between shared memory concept also complicates the problem when the studied model language contains important real-constructs like functions.

This paper presents a new approach for pointer analysis of distributed programs executed on hierarchical memories. The proposed technique has the form of a type system that is simply structured. The presented method is shown on a toy language which is the *while* language enriched with basic commands for pointer manipulations and also enriched with commands vital for distributed execution of programs. The execution model adopted in this paper is the

single program multiply data (*SPMD*) model. In this model the same program is executed on different machines storing different data. Although the language used is simple, it is powerful enough to study distributing and pointer concepts. An abstraction analysis that for a given statement calculates the set of function abstractions that the statement may evaluate to is introduced in this paper. The abstraction analysis is needed in the proposed pointer analysis. The mathematical soundness of all techniques presented in this paper are discussed. The soundness is proved against a new operational semantics presented in this paper.

Motivation

Figure 1 presents a motivating example of our work. The program of the figure consists of two parts; the first part (lines 1 and 2) introduces definitions for statements S_1 and S_2 , and the second part (lines 3, 4, 5 and 6) is the code that uses these definitions. We suppose that the program is executed on a distributed system that has two machines labeled m_1 and m_2 . We also assume that each of the machines has local registers (say x and y) and has local addresses (say $\{a, b\}$ for m_1 and $\{c, d\}$ for m_2). Therefore the set of global variables, $gVar$, is $\{(m_1, x), (m_1, y), (m_2, x), (m_2, y)\}$ and the set of global addresses, $gAdrrs$, is $\{(1, m_1, a), (1, m_1, b), (2, m_1, a), (2, m_1, b), (1, m_2, a), (1, m_2, b), (2, m_2, a), (2, m_2, b)\}$. Loc denotes the union set of global variables and addresses. We consider the parallelism mode *SPMD*. Our paper presents a new technique for analyzing the pointer content of programs like this one. The proposed technique has the advantage, over any existing technique, of associating each analysis result with a correctness proof. The analysis results of the example program are shown in Figure 2 and Figure 3 for machines m_1 and m_2 , respectively.

1. $S_1 = x;$
2. $S_2 = y;$
3. $x := new_1;$
4. $y := new_2;$
5. $y := transmitS_1\text{from}2;$
6. $x := convert(S_2, 2);$

Fig. 1. A motivating example

Contributions

Contributions of this paper are the following:

1. A new abstraction analysis that calculates the set of abstractions that a statement may evaluate-to.
2. A new pointer analysis technique, that is context and flow-sensitive, for distributed programs.

Program point	Pointer information for m_1
the first point	$\{l \mapsto \emptyset \mid l \in Loc\}$
the point between 3 & 4	$\{(m_1, x) \mapsto \{(1, \{m_1\})\}, l \mapsto \emptyset \mid l \neq (m_1, x)\}$
the point between 4 & 5	$\{(m_1, x) \mapsto \{(1, \{m_1\})\}, (m_1, y) \mapsto \{(2, \{m_1\})\},$ $l \mapsto \emptyset \mid l \notin \{(m_1, x), (m_1, y)\}\}$
the point between 5 & 6	$\{(m_1, x) \mapsto \{(1, \{m_1\})\}, (m_1, y) \mapsto \{(2, \{m_1\}), (1, \{m_2\})\},$ $l \mapsto \emptyset \mid l \notin \{(m_1, x), (m_1, y)\}\}$
the last point	$\{(m_1, x) \mapsto \{(1, \{m_1, m_2\})\}, (m_1, y) \mapsto \{(2, \{m_1\}), (1, \{m_2\})\},$ $l \mapsto \emptyset \mid l \notin \{(m_1, x), (m_1, y)\}\}$

Fig. 2. Results of pointer analysis of the program in Figure [1](#) on the machine m_1

Program point	Pointer information for m_2
the first point	$\{l \mapsto \emptyset \mid l \in Loc\}$
the point between 3 & 4	$\{(m_2, x) \mapsto \{(1, \{m_2\})\}, l \mapsto \emptyset \mid l \neq (m_2, x)\}$
the point between 4 & 5	$\{(m_2, x) \mapsto \{(1, \{m_2\})\}, (m_2, y) \mapsto \{(2, \{m_2\})\},$ $l \mapsto \emptyset \mid l \notin \{(m_2, x), (m_2, y)\}\}$
the point between 5 & 6	$\{(m_2, x) \mapsto \{(1, \{m_2\})\}, (m_2, y) \mapsto \{(2, \{m_2\}), (1, \{m_2\})\},$ $l \mapsto \emptyset \mid l \notin \{(m_2, x), (m_2, y)\}\}$
the last point	$\{(m_2, x) \mapsto \{(1, \{m_1, m_2\})\}, (m_1, y) \mapsto \{(2, \{m_1\}), (1, \{m_2\})\},$ $l \mapsto \emptyset \mid l \notin \{(m_2, x), (m_2, y)\}\}$

Fig. 3. Results of pointer analysis of the program in Figure [1](#) on the machine m_2

3. A context insensitive variation for our main context and flow-sensitive pointer analysis.
4. A new operational-semantics for distributed programs on a hierarchy memory model.

Organization

The rest of the paper is organized as follows. Section [2](#) presents our hierarchy memory model. The language that we use to present our analyses and an operational semantics to its constructs are also presented in Section [2](#). Our main analyses are introduced in Section [3](#). These analyses include an abstraction analysis, a pointer analysis that is context and flow-sensitive for distributed programs, and a context insensitive variation of the previous analysis. Related and future work are briefed in Section [4](#).

2 Memory Model, Language, and Operational Semantics

Memory models of parallel computers are typically hierarchical [\[20,21\]](#). In these models each process is often assigned local stores. Caches and local addresses are examples of hierarchies inside processors. The Cell game processor provides

an example where each SPE processor is assigned a local store that is accessible by other SPEs via operations for memory move. It is possible to increase levels of partitions via partitioning a computational grid memory into clusters which in turn can be partitioned into nodes (Figure 4).

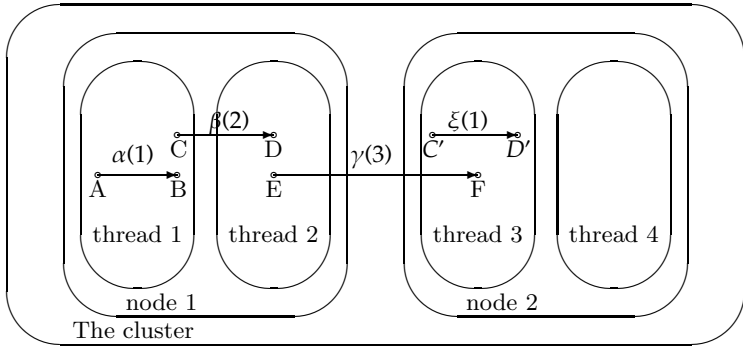


Fig. 4. Grid memory

A memory of two-levels hierarchy is a typical choice for most PGAS [2,16,27]. The two levels are a local one devoted to a particular thread and a shared one serving all threads. The basic idea in PGAS languages is to assign each pointer the memory-hierarchy level that the pointer may reference. The memory model in Figure 4 is a three-levels hierarchy (a cluster, a node, and a thread). For example in Figure 4, pointers α and ξ are thread pointers and they can reference addresses on thread 1 and 4, respectively. However pointers β and γ are node and cluster pointers, respectively. While β can reference addresses on node 1, γ can reference addresses on the cluster. Such domains of pointers can be represented by assigning each pointer a number (width) like edge labels in Figure 4. Clearly the higher the width of the pointer, the more expensive to manipulate the pointer. The manipulation costs of pointers include representation costs, access costs, and dereference costs. The direction in hardware research is to increase hierarchy levels. Therefore it is extremely important for software to benefit from the hierarchy [20,21,2].

We use a simple language [17,18], named $While_d$, to present the results of this paper. $While_d$ is the well-known *while* language enriched with pointer, parallelism, and function constructions [20,21]. The model of parallelism used in $While_d$ is SPMD which means that the same code is executed on all machines. We assume that h denotes the hierarchy height of memory. Consequently, widths of pointers are in the interval $[1, h]$. Figure 5 presents the syntax of $While_d$. The language uses a fixed set of variables, $lVar$, each of which is machine-private. According to the $While_d$ syntax, each program consists of a sequence of definitions followed by a statement ($Defs : S$). The first part of program, $Defs$, is assignments of names to statements. These names can be used in S , the

$$\begin{aligned}
\textit{name} &::= \textit{'string of characters'}. \\
S \in \textit{Stmts} &::= n \mid \textit{true} \mid \textit{false} \mid x \mid S_1 \textit{i}_{op} S_2 \mid S_1 \textit{b}_{op} S_2 \mid *S \mid \textit{skip} \mid \textit{name} \mid x := S \mid S_1 \leftarrow S_2 \mid \\
&S_1; S_2 \mid \textit{if } S \textit{ then } S_t \textit{ else } S_f \mid \textit{while } S \textit{ do } S_t \mid \lambda x.S \mid S_1 S_2 \mid \textit{letrec } x = S \textit{ in } S' \mid \\
&\textit{new}_l \mid \textit{convert } (S, n) \mid \textit{transmit } S_1 \textit{ from } S_2. \\
\textit{Defs} &::= (\textit{name} = S); \textit{Defs} \mid \varepsilon. \\
\textit{Program} &::= \textit{Defs} : S.
\end{aligned}$$

where

$x \in \textit{lVar}$, an infinite set of variables, $n \in \mathbb{Z}$ (integers), $\textit{i}_{op} \in \mathbb{I}_{op}$ (integer-valued binary operations), and $\textit{b}_{op} \in \mathbb{B}_{op}$ (Boolean-valued binary operations).

Fig. 5. The programming language \textit{while}_d

second part of program. For each program of the language \textit{While}_d , our semantics assigns a function fd of the domain $\textit{Function-defs}$:

$$fd \in \textit{Function-defs} = \textit{'strings of characters'} \rightarrow \textit{Stmts}.$$

The map fd is supposed to link each name with its definition. The map fd is built using the following inference rules:

$$\frac{}{\varepsilon : fd \rightsquigarrow fd} (fd_1) \quad \frac{\textit{Defs} : fd[\textit{name} \mapsto S] \rightsquigarrow fd'}{(\textit{name} = S); \textit{Defs} : fd \rightsquigarrow fd'} (fd_2)$$

Therefore for a program $\textit{Defs} : S$, we calculate $\textit{Defs} : \emptyset \rightsquigarrow fd$ using the above inference rules to construct fd .

We define the meaning of \textit{While}_d statements by their operational semantics, i.e., by introducing transition relations \rightsquigarrow_m ; between pairs of statements and states and pairs of values and states. Different components used in the inference rules of the semantics are introduced in the following definition.

- Definition 1.**
1. The set of global variables, denotes by $gVar$, is defined as $gVar = \{(m, x) \mid m \in M, x \in \textit{lVar}\}$.
 2. The set of global addresses, denotes by $gAddrs$, is defined as $gAddrs = \{g = (l, m, a) \mid l \in L, m \in M, a \in \textit{lAddrs}\}$.
 3. $loc \in Loc = gAddrs \cup gVar$.
 4. $v \in \textit{Values} = \mathbb{Z} \cup gAddrs \cup \{\textit{true}, \textit{false}\} \cup \{\lambda x.S \mid S \in \textit{Stmt}\}$.
 5. $\delta \in \textit{States} = Loc \rightarrow \textit{Values}$.

The symbols M, W , and \textit{lAddrs} denote the set of machines labels (integers), the set of width $\{1, \dots, h\}$, and the set of local addresses located on each single machine, respectively. The set of labels of allocation sites is denoted by L .

The semantics produces judgments of the form $(S, \delta) \rightsquigarrow_m (v, \delta')$. The judgement means that executing the statement S on the machine m and in the state δ results

in the value v and modifying the state δ to become δ' . The notation $\delta[x \mapsto v]$ denotes the map $\lambda y. \text{if } y = x \text{ then } v \text{ else } \delta(y)$.

The inference rules of our semantics are as follows:

$$\begin{array}{c}
(n, \delta) \rightsquigarrow_m (n, \delta) \quad (true, \delta) \rightsquigarrow_m (true, \delta) \quad (false, \delta) \rightsquigarrow_m (false, \delta) \quad (x, \delta) \rightsquigarrow_m (\delta(x), \delta) \\
\\
(\lambda x.S, \delta) \rightsquigarrow_m (\lambda x.S, \delta)(abs) \quad \frac{(S_1, \delta) \rightsquigarrow_m (n_1, \delta'') \quad (S_2, \delta'') \rightsquigarrow_m (n_2, \delta')}{(S_1 \text{ i}_{op} S_2, \delta) \rightsquigarrow_m \begin{cases} (n_1 \text{ i}_{op} n_2, \delta'), n_1 \text{ i}_{op} n_2 \in \mathbb{Z}; \\ abort, & \text{otherwise.} \end{cases}} \quad (int-stmt) \\
\\
\frac{(S_1, \delta) \rightsquigarrow_m (b_1, \delta'') \quad (S_2, \delta'') \rightsquigarrow_m (b_2, \delta')}{(S_1 \text{ b}_{op} S_2, \delta) \rightsquigarrow_m \begin{cases} (b_1 \text{ b}_{op} b_2, \delta'), b_1 \text{ b}_{op} b_2 \text{ is a Boolean value;} \\ abort, & \text{otherwise.} \end{cases}} \quad (bo-stmt) \\
\\
\frac{(S, \delta) \rightsquigarrow_m (g, \delta')}{(*S, \delta) \rightsquigarrow_m \begin{cases} (\delta'(g), \delta'), g \in gAddr; \\ abort, & \text{otherwise.} \end{cases}} \quad (de-ref) \quad (skip, \delta) \rightsquigarrow_m (0, \delta) \\
\\
\frac{(S, \delta) \rightsquigarrow_m abort \quad (S, \delta) \rightsquigarrow_m (v, \delta') \quad (S_1, \delta) \rightsquigarrow_m abort}{(x := S, \delta) \rightsquigarrow_m abort \quad (x := S, \delta) \rightsquigarrow_m (v, \delta'[x \mapsto v]) \quad (S_1; S_2, \delta) \rightsquigarrow_m abort} \\
\frac{(S_1, \delta) \rightsquigarrow_m abort \text{ or for } v \notin gAddr. \quad (S_1, \delta) \rightsquigarrow_m (v, \delta'')}{(S_1 \leftarrow S_2, \delta) \rightsquigarrow_m abort} \quad (\leftarrow_1) \\
\\
\frac{(S_1, \delta) \rightsquigarrow_m (v, \delta'') \quad (S_2, \delta'') \rightsquigarrow_m abort}{(S_1, \delta) \rightsquigarrow_m (v, \delta'') \quad (S_2, \delta'') \rightsquigarrow_m abort} \quad (\leftarrow_2) \\
\\
\frac{(S_1, \delta) \rightsquigarrow_m (g, \delta'') \quad (S_2, \delta'') \rightsquigarrow_m (v, \delta'') \quad g \in gAddr}{(S_1 \leftarrow S_2, \delta) \rightsquigarrow_m (v, \delta''[g \mapsto v])} \quad (\leftarrow_3) \quad \frac{(S_1, \delta) \rightsquigarrow_m abort}{(S_1; S_2, \delta) \rightsquigarrow_m abort} \\
\\
\frac{(S_1, \delta) \rightsquigarrow_m (v_1, \delta'') \quad (S_2, \delta'') \rightsquigarrow_m (v_2, \delta') \quad (S_1, \delta) \rightsquigarrow_m (v_1, \delta'') \quad (S_2, \delta'') \rightsquigarrow_m abort}{(S_1; S_2, \delta) \rightsquigarrow_m (v_2, \delta')} \\
\\
\frac{(S, \delta) \rightsquigarrow_m (true, \delta'') \quad (S_t, \delta'') \rightsquigarrow_m abort \quad (S, \delta) \rightsquigarrow_m (true, \delta'') \quad (S_t, \delta'') \rightsquigarrow_m (v, \delta')}{(if S then S_t else S_f, \delta) \rightsquigarrow_m abort \quad (if S then S_t else S_f, \delta) \rightsquigarrow_m (v, \delta')} \\
\\
\frac{(S, \delta) \rightsquigarrow_m (false, \delta'') \quad (S_f, \delta'') \rightsquigarrow_m abort \quad (S, \delta) \rightsquigarrow_m (false, \delta'') \quad (S_f, \delta'') \rightsquigarrow_m (v, \delta')}{(if S then S_t else S_f, \delta) \rightsquigarrow_m abort \quad (if S then S_t else S_f, \delta) \rightsquigarrow_m (v, \delta')} \\
\\
\frac{(if S then S_t else S_f, \delta) \rightsquigarrow_m abort \quad (S, \delta) \rightsquigarrow_m abort}{(if S then S_t else S_f, \delta) \rightsquigarrow_m abort \quad (S, \delta) \rightsquigarrow_m abort} \\
\\
\frac{(if S then S_t else S_f, \delta) \rightsquigarrow_m abort \quad (while S do S_t, \delta) \rightsquigarrow_m abort \quad (S, \delta) \rightsquigarrow_m (false, \delta'') \quad (S, \delta) \rightsquigarrow_m (true, \delta'') \quad (S_t, \delta'') \rightsquigarrow_m abort}{(while S do S_t, \delta) \rightsquigarrow_m (skip, \delta) \quad (while S do S_t, \delta) \rightsquigarrow_m abort} \\
\\
\frac{(S, \delta) \rightsquigarrow_m (true, \delta'') \quad (S_t, \delta'') \rightsquigarrow_m (v'', \delta'') \quad (while S do S_t, \delta'') \rightsquigarrow_m (v', \delta')}{(while S do S_t, \delta) \rightsquigarrow_m (skip, \delta) \quad (while S do S_t, \delta) \rightsquigarrow_m abort} \\
\\
\frac{(S, \delta) \rightsquigarrow_m (true, \delta'') \quad (S_t, \delta'') \rightsquigarrow_m (v'', \delta'') \quad (while S do S_t, \delta'') \rightsquigarrow_m abort}{while S do S_t : (\delta, p) \rightsquigarrow_m (v', \delta')} \\
\\
\frac{(S, \delta) \rightsquigarrow_m (true, \delta'') \quad (S_t, \delta'') \rightsquigarrow_m (v'', \delta'') \quad (while S do S_t, \delta'') \rightsquigarrow_m abort}{(S, \delta) \rightsquigarrow_m (true, \delta'') \quad (S_t, \delta'') \rightsquigarrow_m (v'', \delta'') \quad (while S do S_t, \delta'') \rightsquigarrow_m abort} \\
\\
\frac{(while S do S_t, \delta) \rightsquigarrow_m abort \quad (S_1, \delta) \rightsquigarrow_m (\lambda x.S'_1, \delta'') \quad (S'_1[S_2/x], \delta'') \rightsquigarrow_m (v, \delta')}{(S_1 S_2, \delta) \rightsquigarrow_m (v, \delta')} \quad (appl)
\end{array}$$

$$\begin{array}{c}
\frac{(S, \delta) \rightsquigarrow_m (v, \delta'') \quad (S'[v/x], \delta'') \rightsquigarrow_m (v', \delta')}{(letrec\ x = S\ in\ S', \delta) \rightsquigarrow_m (v', \delta')} \quad (letrec) \quad \frac{a \in lAddr \quad a \text{ is fresh on } m}{(new_l, \delta) \rightsquigarrow_m ((l, m, a), \delta[l, m, a] \mapsto null)} \\
\frac{(S, \delta) \rightsquigarrow_m (g = (l, m', a), \delta') \quad hdist(m, m') \leq n}{(convert(S, n), \delta) \rightsquigarrow_m (g, \delta')} \quad (conv) \quad \frac{(fd(name), \delta) \rightsquigarrow_m v, \delta')}{(name, \delta) \rightsquigarrow_m (v, \delta')} \quad (name) \\
\frac{(S_2, \delta) \rightsquigarrow_m (m', \delta'') \quad m' \in M \quad (S_1, \delta'') \rightsquigarrow_{m'} (v, \delta')}{(transmit\ S_1\ from\ S_2, \delta) \rightsquigarrow_m (v, \delta')} \quad (trans)
\end{array}$$

Some comments on the inference rules are in order. The rules for integer and Boolean statements (*(int-stmt)* and *(bo-stmt)*) and the rule for abstraction are trivial. The rule *(de-ref)* makes sure that the value being dereferenced is indeed a global address. The rules (\leftarrow_1) , (\leftarrow_2) , and (\leftarrow_3) treat the assignment through references. The rules make sure that S_1 is computable and it is a global addresses, otherwise the execution aborts. Every allocation site is assigned a label (the subscription l of *new_l*). Theses labels simplify the problem of pointer analysis. The allocation statement allocates a fresh local address on the machine m and initializes the address to *null*. The rule *(conv)* makes it clear that the statement *convert(e, n)* changes pointer widths. A function, *hdist*, is used in the rule *(conv)* to calculate the distance between machines. Inventing such a function is easy. According to this rule, the change of the width is only allowed if the distance between machines is within the provided range, n . The rule *(trans)* clarifies that the statement *transmit S₁ from S₂* amounts to evaluating the statement S_1 on the machine S_2 and then sending the value to other machines.

Function abstractions and applications are treated in rules *(abs)* and *(app)*, respectively. The rule *(app)* asks for S_1 to evaluate to an abstraction, say $\lambda x.S'_1$. The value of the application is then the result of substituting v (value of S_2) for x in S'_1 . Clearly, the rule applies *call by value* rather than *call by name*. The formal semantics of *letrec* statement amounts to the application specified in the rule *(letrec)*. The intuition of this statement is that it is a well known tool for expressing function recursion.

$$\frac{\theta : \{1, 2, \dots, |M|\} \rightarrow M \quad (S, \delta) \rightsquigarrow_{\theta(1)} (v_1, \delta_1) \rightsquigarrow_{\theta(2)} (v_2, \delta_2) \rightsquigarrow_{\theta(3)} \dots \rightsquigarrow_{\theta(|M|)} (v_{|M|}, \delta_{|M|})}{(Defs : S, \delta) \rightsquigarrow_M (v_{|M|}, \delta_{|M|})} \quad (main-sem)$$

The rule *(main-sem)* provides the semantics of running the program $Defs : S$ on our distributed systems. This rules gives an approximal simulation for executing the program $Defs : S$ using the parallelism model SPMD.

3 Pointer Analysis

This section presents a pointer analysis [10,8,13,9] for the language *while_d*. The proposed technique is both flow and context-sensitive. An adaptation of our technique, towards a flow-sensitive and context-insensitive technique, is also presented. The analysis is presented first for single machines (of set M) and then

a rule that joins results of different machines is presented. The analysis has the form of a type system that assigns to each program point a type capturing points-to information. Types assigned to program points are constructed in a post-type derivation process that starts with the empty set as the type for the program's first point. The derivation of calculating the post type serves as a correctness proof for the analysis results. Such proofs are required in applications like certified code or proof-carrying code. These proofs make the results of analysis certified. Therefore in these applications each pointer analysis is assigned with a proof (type derivation) for the correctness of the analysis results [23].

Now we give intuition of Definition 2. Towards abstracting concrete memory addresses, the concept of abstract address is introduced in the following definition. An abstract address is a pair of an allocation site and a set of machines (subset of M). An order relation is defined on the set of abstract addresses. By this relation, an abstract location a_1 is included in another abstract location a_2 if the two addresses have the same location component and the machines set of a_1 is included in that of a_2 . A set of abstract addresses is *compact* if it does not contain two distinct addresses with the same location. The set of all *compact* subsets of $Addr_s_a$ is denoted by \mathcal{C} (Definition 2.3). A Hoare ordering style on the set \mathcal{C} is introduced in Definition 2.4. Types, *Pointer-types*, of our proposed type system for pointer analysis have the form of maps from $Addr_s_a \cup lVar$ to \mathcal{C} . A point-wise ordering on the set *Pointer-types* is presented in Definition 2.6.

For each of the statements assigned names in the *Defs* part of programs built in the language $while_a$, the function fe (Definition 2.7) stores a pointer effect (type). These pointer effects are calculated by rules (fe_1) and (fe_2) introduced below. The pointer effects capture pointers that executing a statement may introduce to an empty memory. Not far from the reader expectations, pointer effects are important for studying context-sensitive pointer analysis of our language.

A concrete global address $g = (l, m, a)$ is abstracted by an abstract address (l', ms) , denoted by $g = (l, m, a) \models (l', ms)$, if $l = l'$ and $m \in ms$. A concrete state δ is of a pointer type p , denoted by $\delta \models p$, if for every $x \in dom(\delta)$, then if $\delta(x)$ is a global address then $\delta(x)$ is abstracted by an abstract address in $p(x)$.

Definition 2. 1. $Addr_s_a = L \times \mathcal{P}(M)$.

2. $\forall (l_1, ms_1), (l_2, ms_2) \in Addr_s_a. (l_1, ms_1) \leq (l_2, ms_2) \stackrel{\text{def}}{\iff} l_1 = l_2 \text{ and } ms_1 \subseteq ms_2.$

3. $\mathcal{C} = \{S \in \mathcal{P}(Addr_s_a) \mid (l_1, ms_1), (l_2, ms_2) \in S \implies l_1 \neq l_2\}.$

4. $\forall S_1, S_2 \in \mathcal{C}. S_1 \ll S_2 \stackrel{\text{def}}{\iff} \forall x \in S_1. \exists y \in S_2. x \leq y.$

5. $p \in \text{Pointer-types} = Addr_s_a \cup lVar \rightarrow \mathcal{C}.$

6. $\forall p_1, p_2 \in \text{Pointer-types}. p_1 \sqsubseteq p_2 \stackrel{\text{def}}{\iff} \forall x \in dom(p_1). p_1(x) \ll p_2(x).$

7. $fe \in \text{Function-effects} = \text{'strings of characters'} \rightarrow \text{Pointer-types}.$

Definition 3. The concretization and abstraction maps between concrete and abstract addresses are defined as follow:

$$Con : Addr_s_a \rightarrow \mathcal{P}(gAddr_s) : (l, ms) \mapsto \{(l, m, a) \in gAddr_s \mid m \in ms\}.$$

$$Abst : gAddr_s \rightarrow Addr_s_a : (l, m, a) \mapsto (l, \{m\}).$$

Definition 4. – A concrete address $g = (l, m, a)$ is said to be abstracted by an abstract address $a = (l', ms)$, denoted by $g \models a$, iff $l = l'$ and $m \in ms$.

- A concrete state δ is said to be of type p , denoted by $\delta \models p$, iff
 - $\forall x \in lVar. \delta(x) \in gAddr s \implies \exists (l, ms) \in p(x). \delta(x) \models (l, ms)$, and
 - $\forall g \in gAddr s. \delta(g) \in gAddr s \implies ((\forall a. g \models a). \exists a' \in p(a)). \delta(g) \models a'$.

It is not hard to see that the set of pointer types form a complete lattice. Calculating joins in this lattice is an ease exercise which we leave for the interested reader to do.

3.1 Abstraction Analysis

An analysis that for a given statement calculates the set of abstractions that the given statement may evaluate to is required for our pointer analysis. For example, the statement *if* $b > 0$ *then* $\lambda x.u$ *else* $\lambda y.v$ may evaluate to the abstraction $\lambda x.u$ or to the abstraction $\lambda y.v$ depending on the value of b . This section presents an abstraction analysis, an analysis calculating the set of abstractions that a statement may evaluate to. The analysis is achieved via the following set of inference rules:

$$\begin{array}{c}
 \frac{}{n : abs \rightarrow abs} (n) \quad \frac{S_2 : abs \rightarrow abs'}{S_1 \leftarrow S_2 : abs \rightarrow abs'} (:= *^p) \quad \frac{fd(name) : abs \rightarrow abs'}{name : abs \rightarrow abs'} (name^{abs}) \\
 \\
 \frac{S_t : abs \rightarrow abs_t \quad S_f : abs \rightarrow abs_f}{if S then S_t else S_f : abs \rightarrow abs_t \cup abs_f} (if^p) \quad \frac{S_2 : abs \rightarrow abs'}{S_1; S_2 : abs \rightarrow abs'} (seq^p) \\
 \\
 \frac{S_1 : abs \rightarrow abs'' \quad \forall (\lambda x.S_i) \in abs''. S_i[S_2/x] : abs \rightarrow abs_i}{S_1 S_2 : abs \rightarrow \cup_i abs_i} (app_n^p) \\
 \\
 \frac{}{\lambda x.S : abs \rightarrow abs \cup \{\lambda x.S\}} (abs^p) \quad \frac{S : abs \rightarrow abs'}{while S do S_t : abs \rightarrow abs'} (wh^p) \\
 \\
 \frac{(\lambda x.S')S : abs \rightarrow abs'}{letrec x = S in S' : abs \rightarrow abs'} (letrec^p) \quad \frac{S_1 : abs \rightarrow abs'}{transmit S_1 from S_2 : abs \rightarrow abs'} (trans^p)
 \end{array}$$

Some comments on the inference rules are in order. Since integer statements $(n, S_1 \text{ } i_{op} \text{ } S_2)$ do not evaluate to abstractions the rule (n) does not change the input set of abstractions, abs . Other statements that do not evaluate to abstractions (like assignment statement) have inference rules similar to (n) . For a given statement S , we use the inference rules above to find abs' such that $S : \emptyset \rightarrow abs'$. The set abs' contains abstractions that S may evaluate to.

It is straightforward to prove the following lemma:

Lemma 1. *Suppose that $(S, \delta) \rightarrow (v, \delta')$ and $S : \emptyset \rightarrow abs'$. If v is an abstraction, then $v \in abs'$.*

3.2 Context and Flow Sensitive Pointer Analysis

Now we are ready to present the inference rules for our type system for pointer analysis of distributed programs [20,21,24,26,27]. The pointer analysis treated by the following rules is of type flow and context-sensitive. The rules illustrate how different statements update a pointer type, p . Judgement produced by the type system have the form $S : p \rightarrow_m (A, p')$; A contains abstractions for all concrete addresses that may result from executing the statement S in a concrete state of type p . Moreover, if the execution ended then the resulting concrete memory state is of type p' .

$$\begin{array}{c}
\frac{}{n : p \rightarrow_m (\emptyset, p)} \quad \frac{}{x : p \rightarrow_m (p(x), p)} \quad \frac{S : p \rightarrow_m (A, p')}{*S : p \rightarrow_m (\sup\{p'(a) \mid a \in A\}, p')} \quad (* :=^p) \\
\\
\frac{}{skip : p \rightarrow_m (\emptyset, p)} \quad \frac{S : p \rightarrow_m (A, p')}{x := S : p \rightarrow_m (A, p'[x \mapsto A])} \quad (:=^p) \\
\\
\frac{S_1 : p \rightarrow_m (A_1, p_1) \quad S_2 : p_1 \rightarrow_m (A_2, p_2)}{S_1 \leftarrow S_2 : p \rightarrow_m} \quad (\leftarrow^p) \\
\\
\frac{S_1 : p \rightarrow_m (A'', p'') \quad S_2 : p'' \rightarrow_m (A, p')}{S_1; S_2 : p \rightarrow_m (A', p')} \quad (seq^p) \quad \frac{S_t : p \rightarrow_m (A, p') \quad S_f : p \rightarrow_m (A, p')}{if \ S \ then \ S_t \ else \ S_f : p \rightarrow_m (A, p')} \quad (if^p) \\
\\
\frac{S_1 : \emptyset \Rightarrow abs \neq \emptyset \quad \forall \lambda x. S_1' \in abs. \ S_1'[S_2/x] : p \rightarrow_m (A, p')}{S_1 S_2 : p \rightarrow_m (A, p')} \quad (app^p) \\
\\
\frac{fd(name) : p \rightarrow_m (A, p')}{name : p \rightarrow_m (A, p')} \quad (name^p) \quad \frac{S_t : p \rightarrow_m (A, p)}{while \ S \ do \ S_t : p \rightarrow_m (A, p)} \quad (wh^p) \quad \frac{S : p \rightarrow_m (A, p')}{\lambda x. S : p \rightarrow_m (A, p')} \quad (abs^p) \\
\\
\frac{(\lambda x. S')S : p \rightarrow_m (A, p')}{letrec \ x = S \ in \ S' : p \rightarrow_m (A, p')} \quad (letrec^p) \quad \frac{}{new_l : p \rightarrow_m (\{(l, \{m\})\}, p)} \quad (new^p) \\
\\
\frac{S : p \rightarrow_m (A, p')}{convert \ (S, n) : p \rightarrow_m (\{(l, \{m' \in ms \mid hdist(m, m') \leq n\}) \mid (l, ms) \in A\}, p')} \quad (convert^p) \\
\\
\frac{S_2 : p \rightarrow_m (A'', p'') \quad S_1 : p'' \rightarrow_m (A, p')}{transmit \ S_1 \ from \ S_2 : p \rightarrow_m (\{(l, M) \mid (l, ms) \in A\}, p')} \quad (trans^p) \\
\\
\frac{p'_1 \leq p_1 \quad S : p_1 \rightarrow_m (A, p_2) \quad A \subseteq A' \quad p_2 \leq p'_2}{S : p'_1 \rightarrow_m (A', p'_2)} \quad (csq^p) \\
\\
\frac{Defs : \emptyset \rightsquigarrow fd \quad S : p \rightarrow_m (A', p')}{Defs : S : p \rightarrow_m (A', p')} \quad (prg^p)
\end{array}$$

Some comments on the rules are in order. The statements $true$, $false$, $S_1 \ i_{op} \ S_2$ and $S_1 \ b_{op} \ S_2$ have inference rules similar to that of n . The rules for local variables, integer, and Boolean statements are clear; the pointer pre-type does not get updated. The allocation rule, (new^p), results

in the abstract location consists of the allocation site and the number of the machine on which the allocation is taking place. The de-referencing rule, ($* :=^p$), first calculates the set of addresses that S may evaluate to. Then the rule sums the contents of all these addresses. The sequencing rule, (seq^p), is similar to the corresponding semantics rule. The assignment rule, ($:=^p$), copies the abstract addresses, resulted from evaluating the right hand side, into the points-to set of the variable x . As the conversion statement only succeeds if S evaluates to a global address on a machine that is within distance n from m . The rule ($convert^p$) ignores all abstract addresses that are not within the distance n .

This paper considers the parallelism model SPMD (applied in many languages) [2,16,27]. In this model, the *transmit* statement succeeds only if the target statement (S_1) evaluates to abstract addresses that have same cite labels on the current and target machines. Since the distance between these machines is not known statically, the abstract addresses calculated by the rule ($trans^p$) assumes maximum possible distance. For the statement $S_1 \leftarrow S_2$, assignment via references, we illustrate the following example. Let S_1 be a variable r whose points-to set contains the abstract address $a_1 = (l_1, ms_1)$. Let S_2 be a variable s whose points-to set contains the abstract address $a_2 = (l_2, ms_2)$. Then processing the rule must include augmenting the points-to set of a_1 with a_2 . However this is not all; since our parallelism [20,21] model is SPMD, this assignment has to be considered on the other machines, as well. Hence the set $\{(l_2, ms'_1 \cup ms_1 \cup ms_2)\}$ is added to the points-to set of all addressers (l_1, ms'_1) whose first component is l_1 .

The recursion rule, ($letrec^p$), is similar to the corresponding semantics rule. The abstraction rule, (abs^p), is straightforward. The function application rule, (app^p), uses the abstraction analysis presented earlier to calculate the set of abstractions that S_1 may evaluate-to. For each of the calculated abstractions, the rule does the application and achieve the pointer analysis. Finally the obtained post-pointer-types for all abstractions are summed in the post-pointer-type of the application statement.

The following rule calculates the pointer information for running a statement S on all the machine in M using the SPMD model.

$$\frac{\forall m \in M. S : \sup\{p, p_j \mid j \neq i\} \rightarrow_m (A_m, p_m)}{S : p \rightarrow_M (\cup_i A_i, \sup\{p_1, \dots, p_n\})} \text{ (main-pt)}$$

The following theorem proves the soundness of our pointer analysis on each single machine of our $|M|$ machines. The soundness of the analysis for the whole distributed system is inferred in the next corollary.

Theorem 1. *Suppose that $(S, \delta) \rightsquigarrow_m (v, \delta')$, $S : p \rightarrow_m (A', p')$, and $\delta \models p$. Then*

1. $v \in g\text{Addr}s \implies v$ is abstracted by some $a \in A'$, and
2. $\delta' \models p'$.

Proof. The proof is by structure induction on the type derivation. Some cases are detailed below.

- The case of (x^p) : in this case $v = \delta(x)$, $\delta' = \delta$, $A' = p(x)$, and $p' = p$. If $\delta(x)$ is an address, then it is abstracted by some $a \in p(x)$ because $\delta \models p$. Clearly in this case $\delta' \models p'$.
- The case of $(:=^p)$: in this case there exists S'' , δ'' and p'' such that $S = x := S''$, $(S'', \delta) \rightsquigarrow_m (v, \delta'')$, and $S'' : p \rightarrow_m (A', p'')$. Moreover, in this case, $\delta' = \delta''[x \mapsto v]$ and $p' = p''[x \mapsto A']$. Therefore by induction hypothesis on $S'' : p \rightarrow_m (A', p'')$, we conclude that $\delta'' \models p''$ and that if $v \in g\text{Addrs}$ then v is abstracted by some $a \in A'$. These two results together with definitions of p' and δ' imply that $\delta' \models p'$ which completes the proof of this case.
- The case of $(* :=^p)$: in this case there exist $g = (l, m, a) \in g\text{Addrs}$, S'' , and A'' such that $S = *S''$, $(S'', \delta) \rightsquigarrow_m (g, \delta')$, and $S'' : p \rightarrow_m (A'', p')$. Moreover, in this case, $A' = \sup\{p'(a) \mid a \in A''\}$ and $v = \delta'(g)$. Hence by induction hypothesis on $S'' : p \rightarrow_m (A'', p')$, we conclude that g is abstracted by some $a = (l, ms) \in A''$ and that $\delta' \models p'$. Hence $m \in ms$. Now we suppose that $\delta'(g) \in g\text{Addrs}$ and prove that $\delta'(g)$ is abstracted by some element in A' . Since $\delta' \models p'$, $\delta'(g)$ is abstracted by some element in $p'(a) \subseteq A'$.
- The case of (\leftarrow^p) : in this case there exist $g \in g\text{Addrs}$, S_1, S_2, p_1, p_2, A_1 , and A_2 such that $S = S_1 \leftarrow S_2$, $S_1 : p \rightarrow_m (A_1, p_1)$, $S_2 : p_1 \rightarrow_m (A_2, p_2)$, $(S_1, \delta) \rightsquigarrow_m (g, \delta')$, and $(S_2, \delta'') \rightsquigarrow_m (v', \delta''')$. Moreover, in this case, $\delta' = \delta'''[g \mapsto v']$, $A' = A_2$, and $p' = p_2[(l_1, ms'_1) \mapsto p_2((l_1, ms'_1)) \sqcup \{(l_2, ms'_1 \cup ms_1 \cup ms_2) \mid (l_1, ms_1) \in A_1, (l_2, ms_2) \in A_2\}]$. Hence by induction hypothesis on $S_1 : p \rightarrow_m (A_1, p_1)$ we conclude that g is abstracted by some $(l_1, ms_1) \in A_1$ and that $\delta'' \models p_1$. Again by induction hypothesis on $S_2 : p_1 \rightarrow_m (A_2, p_2)$, we conclude that v is abstracted by some $(l_2, ms_2) \in A_2$ and that $\delta''' \models p_2$. It remains to show that $\delta' \models p'$. By definitions of δ' and p' it is enough to show that if v is a concrete address, then the image of any abstraction of g under p' contains an abstraction to v . But any abstraction of g has allocation site l_1 and an abstraction of v exists in A_2 . Therefore in p' all images of abstract addresses with allocation sites l_1 are augmented with all abstract addresses of A_2 with allocation site l_2 . This augmentation guarantees the required.
- The case of (app^p) : in this case there exist S_1 and S_2 such that $S = S_1 S_2$, $(S_1, \delta) \rightsquigarrow_m (\lambda x.S'_1, \delta'')$, $(S_2, \delta'') \rightsquigarrow_m (v', \delta''')$, and $(S'_1[v'/x], \delta''') \rightsquigarrow_m (v, \delta')$. Moreover, in this case, $A' = A$, $S_1 : \emptyset \Rightarrow \text{abs} \neq \emptyset$ and $\forall \lambda x.S'_1 \in \text{abs}(S'_1[S_2/x] : p \rightarrow_m (A, p'))$. By Lemma [□](#), $\lambda x.S'_1 \in \text{abs}$. Hence $S'_1[S_2/x] : p \rightarrow_m (A, p')$. Therefore by induction hypothesis on $\lambda x.S'_1 \in \text{abs}$, we conclude that $\delta' \models p'$ and that if $v \in g\text{Addrs}$ then v is abstracted by some $a \in A'$.
- The case of $(convert^p)$: in this case there exists S' such that $S = \text{convert}(S', n)$, $(S, \delta) \rightsquigarrow_m (g = (l, m', a), \delta')$, $\text{hdist}(m, m') \leq n$, and $S' : p \rightarrow_m (A, p')$. Moreover, in this case, $A' = \{(l, \{m' \in ms \mid \text{hdist}(m, m') \leq n\}) \mid (l, ms) \in A\}$. By induction hypothesis on $S' : p \rightarrow_m (A, p')$, we conclude that $\delta' \models p'$ and that g is abstracted by some $a = (l', ms') \in A$. As $\text{hdist}(m', m') = 0 \leq n$, $m' \in ms'$, and $a = (l', ms') \in A$, we conclude that g is abstracted by some element in A which completes the proof for this case.
- The case of $(trans^p)$: in this case there exist S_1 and S_2 such that $S = \text{transmit } S_1 \text{ from } S_2$, $S_2 : p \rightarrow_m (A'', p'')$, $S_1 : p'' \rightarrow_m (A, p')$, $(S_2, \delta) \rightsquigarrow_m$

$(m', \delta''), m' \in M$, and $(S_1, \delta'') \rightsquigarrow_{m'} (v, \delta')$. Moreover, in this case, $A' = \{(l, M) \mid (l, m) \in A\}$. By induction hypothesis on S_1 and S_2 , we conclude that $\delta' \models p'$ and that if $v \in gAddr_s$ then it is abstracted by some $a = (l', ms') \in A$. We assume that $v = (l, m, a) \in gAddr_s$. Then $(l, ms) \in A$, for some ms that contains m . We conclude that $(l, M) \in A'$. But this last element abstracts v . This completes the proof for this case.

The following corollary follows from Theorem 1 by using the semantics and pointer rules (*main-sem*) and (*main-pt*), respectively. It is noticeable that subscriptions of arrows in Theorem 1 and Corollary 1 are different.

Corollary 1. *Suppose that $(S, \delta) \rightsquigarrow_M (v, \delta')$, $S : p \rightarrow_M (A', p')$ and $\delta \models p$. Then*

1. $v \in gAddr_s \implies v \in A$, and
2. $\delta' \models p'$.

3.3 Flow Sensitive and Context Insensitive Pointer Analysis

Letting the following rules replace their corresponding ones in the type system of pointer analysis presented above results in a flow sensitive and context insensitive pointer analysis.

$$\begin{array}{c}
 \frac{\text{name} \in \text{dom}(fe)}{\text{name} : p \rightarrow_m \text{sup}\{p, fe(\text{name})\}} \text{(name}^e_1) \\
 \\
 \frac{\varepsilon : fe \rightsquigarrow_m fe \quad (fe_1) \quad \frac{S : \emptyset \rightarrow p \quad \text{Defs} : fe[\text{name} \mapsto p] \rightsquigarrow_m fe'}{\text{(name} = S); \text{Defs} : (fd, fe) \rightsquigarrow_m (fd', fe')} (fe_2)}{\text{Defs} : \emptyset \rightsquigarrow_m fd \quad \text{Defs} : \emptyset \rightsquigarrow_m fe \quad S : p \rightarrow_m p'} (prg^i)}{\text{Defs} : S : p \rightarrow_m p'}
 \end{array}$$

4 Related and Future Work

The language that is studied in the current paper is a generalization of those described in [17,18]. The work in [17] introduces a flow-insensitive pointer analysis for programs sharing memory and running on parallel machines that are hierarchical. Beside making the results vague (inaccurate), the insensitivity of the analysis in [17] forces the pointer analysis to ignore the distributivity of targeted programs which is a major drawback. Also the analysis in [17] does not treat context-sensitivity, which is a basic aspect in real-life programming. The current paper overcomes these drawbacks. Using a two-level hierarchy, in [19], constraint-based analyses to calculate sharing properties and locality information of pointers are introduced. These constraint-based analyses are extensions of the earlier work in [18].

Pointer analysis, that dates back to work in [1], was extended to cover parallel programs. In [23] a pointer analysis that is flow-sensitive, context-sensitive, and

thread-aware is introduced for the Cilk multithreaded programming language. Another pointer analysis that is a mix of flow-sensitive and flow-insensitive for multithreaded languages is introduced in [8]. Examples of flow-insensitive pointer analyses for multithreaded programs are [15,28]. Probabilistic points-to analysis is studied in [4,9]. In [4], using the algorithmic style, the probability that a points-to relationship holds is represented by a quantitative description. On the other hand [9] uses type systems to provide an analysis that very suitable to certified codes or proof carrying software. However pointer analysis, that provides a proof for each pointer analysis, for distributed programs running on hierarchical machines are not considered by any of these analyses.

The importance of distributed programs makes analyzing them the focus of much research activities [20,21,24,16,27]. The concurrent access of threads of a multi-threaded process to a physically distributed memory may result in data racing bugs [5,22]. In [5] a scheme, called DRARS, is introduced for avoidance and replay of this data race. On DSM or multi-core systems, DRARS assists debugging parallel programs. An important issue to many distributed systems applications is the capturing and examining of the concurrent and causal relationships. In [25], an inclusive graph, POG, of the potential behaviors of systems is produced via an analysis that considers the source code of each process. In [26] and on the model of message sending of distributed programs, the classical problems of algorithmic decidability and satisfiability decidability are studied. In this study, communicating through buffers are used to represent distributed programs.

Associating the result of each pointer analysis with a correctness proof is important and required by applications like certified code or proof carrying code. One advantage of the work presented in this paper over any other related work is the constructions of these proofs. The proofs constructed in our approach have the form of a type derivation and this adds to the value of using type systems. Examples of other analyses that have the form of type systems are [10,8,13,9].

Mathematical domains and maps between domains can be used to mathematically represent programs and data structures. This representation is called denotational semantics of programs [3,14,24]. One of our directions for future research is to translate concepts of data and program slicing to the side of denotational semantics [12,7]. Doing so provide a good tool to mathematically study in deep heap slicing. Then obtained results can be translated back to the side of programs and data structures.

References

1. Amme, W., Zehendner, E.: A/D Graphs - a Data Structure for Data Dependence Analysis in Programs with Pointers. In: Böszörme'nyi, L. (ed.) ACPC 1996. LNCS, vol. 1127, pp. 229–230. Springer, Heidelberg (1996)
2. Barpanda, S.S., Mohapatra, D.P.: Dynamic slicing of distributed object-oriented programs. IET Software 5(5), 425–433 (2011)
3. Cazorla, D., Cuartero, F., Ruiz, V.V., Pelayo, F.L.: A denotational model for probabilistic and nondeterministic processes. In: Lai, T.-H. (ed.) ICDCS Workshop on Distributed System Validation and Verification, pp. E41–E48 (2000)

4. Chen, P.-S., Hwang, Y.-S., Ju, R.D.-C., Lee, J.K.: Interprocedural probabilistic pointer analysis. *IEEE Trans. Parallel Distrib. Syst.* 15(10), 893–907 (2004)
5. Chiu, Y.-C., Shieh, C.-K., Huang, T.-C., Liang, T.-Y., Chu, K.-C.: Data race avoidance and replay scheme for developing and debugging parallel programs on distributed shared memory systems. *Parallel Computing* 37(1), 11–25 (2011)
6. El-Zawawy, M., Daoud, N.: New error-recovery techniques for faulty-calls of functions. *Computer and Information Science* 4(3) (May 2012)
7. El-Zawawy, M.A.: Semantic spaces in Priestley form. PhD thesis, University of Birmingham, UK (January 2007)
8. El-Zawawy, M.A.: Flow Sensitive-Insensitive Pointer Analysis Based Memory Safety for Multithreaded Programs. In: Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., Apduhan, B.O. (eds.) ICCSA 2011, Part V. LNCS, vol. 6786, pp. 355–369. Springer, Heidelberg (2011)
9. El-Zawawy, M.A.: Probabilistic pointer analysis for multithreaded programs. *ScienceAsia* 37(4) (December 2011)
10. El-Zawawy, M.A.: Program optimization based pointer analysis and live stack-heap analysis. *International Journal of Computer Science Issues* 8(2) (March 2011)
11. El-Zawawy, M.A.: Dead code elimination based pointer analysis for multi-threaded programs. *Journal of the Egyptian Mathematical Society* (January 2012), doi:10.1016/j.joems.2011.12.011
12. El-Zawawy, M.A., Jung, A.: Priestley duality for strong proximity lattices. *Electr. Notes Theor. Comput. Sci.* 158, 199–217 (2006)
13. El-Zawawy, M.A., Nayel, H.A.: Partial redundancy elimination for multi-threaded programs. *IJCSNS International Journal of Computer Science and Network Security* 11(10) (October 2011)
14. Guo, M.: Denotational semantics of an hpf-like data-parallel language model. *Parallel Processing Letters* 11(2/3), 363–374 (2001)
15. Hicks, J.: Experiences with compiler-directed storage reclamation. In: FPCA, pp. 95–105 (1993)
16. Seragiotto Jr., C., Fahringer, T.: Performance analysis for distributed and parallel java programs with aksum. In: CCGRID, pp. 1024–1031. IEEE Computer Society (2005)
17. Kamil, A., Yelick, K.A.: Hierarchical Pointer Analysis for Distributed Programs. In: Riis Nielson, H., Filé, G. (eds.) SAS 2007. LNCS, vol. 4634, pp. 281–297. Springer, Heidelberg (2007)
18. Liblit, B., Aiken, A.: Type systems for distributed data structures. In: POPL, pp. 199–213 (2000)
19. Liblit, B., Aiken, A., Yelick, K.A.: Type Systems for Distributed Data Sharing. In: Cousot, R. (ed.) SAS 2003. LNCS, vol. 2694, pp. 273–294. Springer, Heidelberg (2003)
20. Lindberg, P., Leingang, J., Lysaker, D., Khan, S.U., Li, J.: Comparison and analysis of eight scheduling heuristics for the optimization of energy consumption and makespan in large-scale distributed systems. *The Journal of Supercomputing* 59(1), 323–360 (2012)
21. Onbay, T.U., Kantarci, A.: Design and implementation of a distributed teleradiology system: Dipacs. *Computer Methods and Programs in Biomedicine* 104(2), 235–242 (2011)
22. Park, C.-S., Sen, K., Hargrove, P., Iancu, C.: Efficient data race detection for distributed memory parallel programs. In: Lathrop, S., Costa, J., Kramer, W. (eds.) SC, p. 51. ACM (2011)

23. Rugina, R., Rinard, M.C.: Pointer analysis for multithreaded programs. In: PLDI, pp. 77–90 (1999)
24. Schwartz, J.S.: Denotational Semantics of Parallelism. In: Kahn, G. (ed.) *Semantics of Concurrent Computation*. LNCS, vol. 70, pp. 191–202. Springer, Heidelberg (1979)
25. Simmons, S., Edwards, D., Kearns, P.: Communication analysis of distributed programs. *Scientific Programming* 14(2), 151–170 (2006)
26. Toporkov, V.V.: Dataflow analysis of distributed programs using generalized marked nets. In: *DepCoS-RELCOMEX*, pp. 73–80. IEEE Computer Society (2007)
27. Truong, H.L., Fahringer, T.: Soft Computing Approach to Performance Analysis of Parallel and Distributed Programs. In: Cunha, J.C., Medeiros, P.D. (eds.) *Euro-Par 2005*. LNCS, vol. 3648, pp. 50–60. Springer, Heidelberg (2005)
28. Zhu, Y., Hendren, L.J.: Communication optimizations for parallel c programs. *J. Parallel Distrib. Comput.* 58(2), 301–332 (1999)

An Approach to Measure Understandability of Extended UML Based on Metamodel

Yan Zhang^{1,2}, Yi Liu^{3,4}, Zhiyi Ma^{3,4}, Xuying Zhao¹,
Xiaokun Zhang¹, and Tian Zhang^{2,5,*}

¹ Department of Computer Science and Technology,
Beijing Electronic Science and Technology Institute, Beijing, P.R. China 100070
zhangyan@besti.edu.cn

² State Key Lab. for Novel Software Technology, Nanjing University,
Nanjing, P.R. China 210093

³ Institute of Software, School of Electronics Engineering and Computer Science,
Peking University, Beijing, P.R. China 100871

⁴ Key Laboratory of High Confidence Software Technologies (Peking University),
Ministry of Education, Beijing, P.R. China 100871
{liuyi07,mzy}@sei.pku.edu.cn

⁵ Department of Computer Science and Technology, Nanjing University,
Nanjing, P.R. China 210093
ztluck@nju.edu.cn

Abstract. Since UML does not provide any guidance for users to select a proper extension pattern, users are not able to assure the quality of extended UMLs, such as understandability, when they focus on their expression power. A metric of understandability for extended UMLs is proposed, which bases on measuring the deviation of understandability between the extended UMLs and the standard UML in their metamodel level. Our proposal can be used to compare different extended UMLs with the same expression power on the understandability characteristic. Moreover, the proposal can guide users to select an appropriate extension pattern to achieve their goal. We give the definition of the metric of understandability and the empirical validation of the proposed metric. A case from a real project is used to explain the application of the proposed metric.

Keywords: understandability, metamodel, UML, extension pattern, software metrics.

1 Introduction

1.1 Problem

Modeling is a primary activity in the software development. Object Management Group (OMG) has proposed Unified Modeling Language (UML) as an universal description language of models. After OMG released UML version 1.1 [1] as the

* Corresponding author.

available specification, UML has been widely studied and used in the academia and the industry. Hitherto, UML has evolved from version 1.1 to version 2.1 [2] and become a de facto standard modeling language used to communicate among software developers.

Although UML is a powerful modeling language, it cannot meet all modeling requirements in the real life because of the diversity of applications. For this reason, users often need to extend UML to meet their specific modeling requirement and make the extended UML better describe a system to be built. With the increasing popularity of UML, extending UML has been a prevalent activity, especially in some application domains such as embedded system and communication.

Besides some built-in extension mechanisms (e.g., stereotype [3], first-class mechanism and profile extension mechanism [4]) in UML, OMG does not give any guidance that can help users to extend UML with a proper *extension pattern*, i.e., a modification of UML metamodel that holds some constraint, to meet a specific modeling requirement. As a result, users may only focus on the expression power of the extended UML but cannot assure its quality, such as understandability, maintainability and usability, when they extend UML for their goal. In this paper we mainly study the understandability of an extended UML. In other words, we focus on the understandability of a modeling language, but not a model, which is described by some modeling language.

1.2 Motivation

Usually, the extension of UML is conducted in the metamodel level of UML, that is, UML is extended by modifying its metamodel, for example, adding some model elements or some relationships [5,6,7]. We notice that the extended UMLs obtained by different extension patterns could have different understandability even though they have the same modeling power. Because the metamodel of UML specifies the syntax of UML, the understandability of UML can be reflected by the understandability of its metamodel.

Based on the observation that more intricate the syntax of a language is, the more difficult is it to learn or use the language, which means that the understandability of the language is worse, we propose a method for measuring the understandability of extended UMLs by a measurement of their metamodels. This method can be used to evaluate various extended UMLs with the same modeling power on the understandability characteristic. Furthermore, it can guide users to select more appropriate extension patterns to achieve their goals of modeling. The kernel of our method is that: by using the understandability of the standard UML metamodel (U_0) as a base, i.e., supposing that the standard UML has been fully understood by an individual, we measure the *deviation of understandability* (ΔU) between the metamodel of standard UML and the metamodel of extended UML, furthermore obtain the understandability of the extended UML metamodel ($U = U_0 + \Delta U$) that reflects the understandability of the extended UML for the same individual. The deviation of understandability is used as an indicator to evaluate different extended UMLs. The bigger the deviation is, the

worse is the understandability of an extended UML, which means that users need more effort to understand it.

In the cognition perspective, the different cognitive subjects that face the same cognitive object would have different cognitive results. Therefore, the ease of understanding the same thing is different for different people. In our proposal, we do not measure the understandability of UML in general sense, but measure the understandability for a specific individual. In other words, we measure the effect on the understandability for the same cognitive subject after the original that had been understood by the cognitive subject is modified partially. The deviation of understandability is used to eliminate the influence on the understandability metric, which results from the human cognitive diversity.

We give a metric definition of understandability for an extended UML and the empirical validation of the proposed metric. Additionally, we apply the proposed metric to evaluate two extended UMLs with the same modeling power in a real project.

The remainder of this paper is organized as follows. Section 2 considers related research works. Section 3 introduces the metamodel of UML and the extension patterns of UML. Section 4 defines the understandability metric for an extended UML and gives the empirical validation of the proposed metric. Section 5 discusses the application of our proposed metric in the evaluation of different extension patterns of UML. Finally, in Section 6 we conclude this paper and discuss the future work.

2 Related Works

Although a few researches conducted the metric of understandability of models [8,9,10,11], they mainly focus on measuring the understandability of a system model that is described by UML. For example, Genero and his colleagues [8,9,10] studied the metric of understandability and modifiability about UML class diagram. They gave the metric definitions, which were composed of the measurement of size and structural complexity of UML class diagram, and the empirical validation of their metric. The main difference between our research and theirs is that our research measures the understandability of a modeling language, which corresponds to M2 level of the four-layered metamodel architecture [12], but their research measures the understandability of a system model, which corresponds to M1 level of the four-layered metamodel architecture. Our research concerns the quality improvement of a modeling language, but their research concerns the quality improvement of a system. Although our research refers many results in [8,9,10], in particular, the controlled experiments in [8] and [10], we do not directly use the size measurement of generalizations and compositions in our metric definition. We advance their contribution to understandability into the computation of the understandability of every class.

Jiang et al. [13] indicated the effect made by different extension patterns on the quality of UML. They defined four-leveled metamodel extension and qualitatively discussed their influence on the readability and expression capability of

the extended UML. Based on their research, this paper quantitatively studies the extension patterns' influence on the understandability of the extended UML.

Ma and his colleagues [14] had given an approach based on OO metrics of metamodel to evaluate the quality of different UML versions, which involves the understandability. They used some design properties, such as design size, abstraction, coupling, polymorphism and complexity, to define the metric of understandability and each design property was mapped to some architectural metrics, e.g., the average number of ancestors, the number of abstract meta-classes, the average number of stereotypes and so on. According to their metrics, the understandability of UML1.1, UML1.4 and UML2.0 are -0.99, -2.02 and -2.87 respectively. Although the trend of the changes in the understandability of UML1.1, UML1.4 and UML2.0 that is reflected by their metrics is same as ours, we disagree with the fact that they assigned the same weight to each design property in their metric definition. Additionally, a big difference between their metric and ours is that our metric evaluates the understandability of different UML versions by the deviation of understandability, i.e., the influence on the understandability corresponding to different extension patterns, rather than by directly measuring the understandability for every UML version. Significantly, our metric can indicate for user the best one among all candidates of extension pattern, but their metric has not the ability.

3 Background

In this section, we briefly introduce UML metamodel and different extension patterns of UML in its metamodel level.

3.1 Metamodel of UML

A modeling language's metamodel provide the constructs, syntax, and semantics for the language [15]. Every element of a modeling language is an instance of certain element in its metamodel [15]. Specifically, UML metamodel defines the abstract syntax of UML. In essence, the metamodel of UML can be considered as a UML class diagram [4]. We formalize the UML metamodel as follows.

UML metamodel $M = \langle C, A, AS, CO, GE \rangle$ is a 5-tuple, where C is the set of classes, A is the set of attributes, AS is the set of association relationships, CO is the set of composition relationships and GE is the set of generalization relationships. For any class $c \in C$, $A^c \subseteq A$ denotes the set of attributes in class c . For any two classes $c_1, c_2 \in C$, $(c_1, c_2) \in CO$ represents that c_2 is a part of c_1 , and c_1 is called as the whole-class (of c_2) and c_2 is called as the part-class (of c_1). For any two classes $c_1, c_2 \in C$, $(c_1, c_2) \in GE$ represents that c_2 inherits c_1 , and c_1 is called as the superclass (of c_2) and c_2 is called as the subclass (of c_1).

Fig. 1 shows the main body of the metamodel of the class defined in UML1.1. It represents that a class includes none or more attributes, operations and methods (which implement some operations), and none or more relationships such as associations and generalizations may exist among classes.

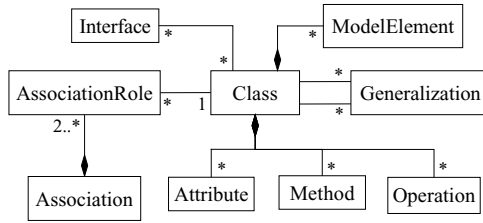


Fig. 1. The metamodel of the class in UML

3.2 Extension Patterns of UML

The syntax of an extended UML, i.e., its metamodel, can be obtained by modifying the metamodel of a standard UML. There are some approaches to modifying UML metamodel and each one hold some constraints. We call them extension patterns shown as follows [13]:

- directly modifying the elements in the standard UML metamodel, such as adding new attributes in original classes (see Fig. 2(a)) or adding new relationships among original classes (see Fig. 2(b));
- directly adding new classes to the standard UML metamodel (see Fig. 2(c));
- adding new classes to the standard UML metamodel by generalizations, i.e., the added classes inherit some original classes (see Fig. 2(d));
- adding new relationships between an added class and an original class (see Fig. 2(e)) or between two added classes (see Fig. 2(f)).

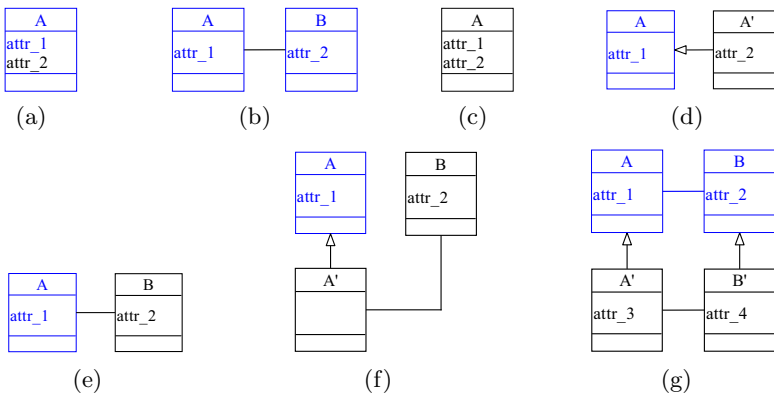


Fig. 2. The extension patterns of UML. The blue color represents the original elements and the black color represents the added elements.

Note that from the viewpoint of the users, the same effect can be achieved by different extension patterns. For example, Fig. 2(a) and Fig. 2(d) have the

same effect, i.e., the modified A and A' can both meet the user's need. Similarly, although the extension patterns are different, the effect of Fig. 2(e) and Fig. 2(f) are the same in users' viewpoint. However, different extension patterns have different effects on the understandability of the extended UMLs.

4 Measurement of Understandability for Extended UML

4.1 Basic Idea

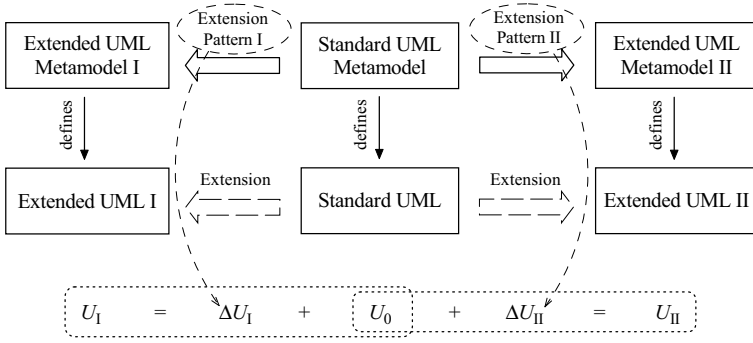
Because UML metamodel amounts to the syntax definition of UML, that is, the metamodel specifies what model elements and relationships among them should exist in UML, the understandability of UML can be defined as *the ease degree of understanding the metamodel of UML*. Generally, UML is extended by adding some new model elements (e.g., classes and attributes) or some new relationships (e.g., associations and generalizations) in UML metamodel. Thus, the definition of an extended UML (i.e., the extended UML metamodel) is a modified metamodel of the standard UML. If U_0 denotes the understandability of the standard UML metamodel, the understandability of an extended UML metamodel, denoted as U , can be defined by $U = U_0 + \Delta U$, where ΔU is the deviation of understandability and it represents the effort increase for re-understanding the modified metamodel. Since any extension of UML is made on the standard UML metamodel, for arbitrary two extended UMLs with the same modeling power, the difference of their understandability, i.e., $U_I = U_0 + \Delta U_I$ and $U_{II} = U_0 + \Delta U_{II}$, results from the difference of ΔU_I and ΔU_{II} . The reason of this difference is the different extension patterns of UML. In other words, different extension patterns lead to different deviations of understandability, furthermore, different understandability of extended UMLs. Therefore, the measurement of understandability for extended UMLs can be solved if we can give the metric definition of the deviation of understandability (i.e., ΔU). Fig. 3 shows the basic ideas about how to measure the understandability of extended UMLs.

There are three kinds of modifications of the metamodel for extending UML: modifying (i.e., adding, deleting or changing) some classes, modifying some attributes in original classes and modifying some relationships. For defining the metric of ΔU , we must find the relation between the change of understandability and every kind of modification. Based on the relation, we will give the metric definition of understandability in the following subsection.

4.2 Metric Definition

Suppose that the standard UML metamodel is $M_0 = \langle C_0, A_0, AS_0, CO_0, GE_0 \rangle$ and the extended UML metamodel is $M_1 = \langle C_1, A_1, AS_1, CO_1, GE_1 \rangle$. Let NA^c be the number of attributes in class c and ΔNA^c be the number of modified attributes in class c of the extended UML metamodel different to that of the standard UML metamodel, which is defined as:

$$\Delta NA^c = \begin{cases} |A_1^c \setminus A_0^c| + |A_0^c \setminus A_1^c|, & \text{if } c \in C_1 \wedge c \in C_0 \\ |A_1^c|. & \text{if } c \in C_1 \wedge c \notin C_0 \end{cases} \quad (1)$$



where U_0 , U_I and U_{II} denote the understandability of the standard UML, extended UML I and extended UML II respectively;
 ΔU_I and ΔU_{II} denote the deviations of understandability caused by extension pattern I and II respectively.

Fig. 3. The schematic diagram of measuring the understandability for extended UMLs

Let ΔU^c and ΔU^a be the deviation of understandability of some class c and some association a after some extension for the standard UML metamodel respectively. According to the study of Briand and Wüst [16], the structural properties, such as size and complexity, affect the understandability. Similarly, the study of Genero et al. [8,9,10] points that structural complexity and size metrics are strongly correlated with the understandability of a class diagram. Based on these conclusions, we give the definition of ΔU^c and ΔU^a as following:

- For any $c \in C_1$, if there does not exist $c' \in C_1$ such that $(c, c') \in GE_1$ or $(c, c') \in CO_1$, then

$$\Delta U^c = \begin{cases} 1, & c \notin C_0 \wedge A^c = \emptyset \\ \alpha \cdot \Delta NA^c, & \text{otherwise} \end{cases} \quad (2)$$

where α is the impact factor of the number of attributes to understandability.

Equation (2) represents that for a class c that does not relate with other classes by generalizations or compositions (see class A in Fig. 2(a) and Fig. 2(c) and class B in Fig. 2(e) and Fig. 2(f)), the number of increased attributes in c (i.e., ΔNA^c) mainly contributes to the change of the ease for re-understanding the modified c . Specifically, if a new class c is empty, i.e., there is no attribute in c , we consider that one unit effort must be taken to understand c .

- For any $c \in C_1$, if there exist $C'_1 \subset C_1$ such that for all $c' \in C'_1$ there is $(c', c) \in GE_1$ and for all $c' \in C_1 \setminus C'_1$ there is $(c', c) \notin GE_1$, then

$$\Delta U^c = \alpha \cdot \Delta NA^c + \beta \sum_{c' \in C'_1} \Delta U^{c'}, \quad (3)$$

where β is the impact factor of the generalization relationships to understandability.

Equation (3) means that for a class c (as a subclass) that relates with some classes (as superclasses) by generalizations (see class A' in Fig. 2(d)), besides the number of increased attributes in c , the change of every superclass's understandability (i.e., $\Delta U^{c'}$) also contributes to the change of the ease for re-understanding the modified c . Since there is an inheritance relationship between a subclass and its every superclass, the ease of understanding the subclass bases on the ease of understanding its superclasses. Thus, the change of understandability of every superclass will influence the understandability of the subclass.

- For any $c \in C_1$, if there exist $C'_1 \subset C_1$ such that for all $c' \in C'_1$ there is $(c, c') \in CO_1$ and for all $c' \in C_1 \setminus C'_1$ there is $(c, c') \notin CO_1$, then

$$\Delta U^c = \alpha \cdot \Delta NA^c + \gamma \sum_{c' \in C'_1} \Delta U^{c'}, \quad (4)$$

where γ is the impact factor of the composition relationships to understandability.

Equation (4) means that for a class c (as a whole-class) that relates with some classes (as part-classes) by compositions, besides the number of increased attributes in c , the change of every part-class's understandability (i.e., $\Delta U^{c'}$) also contributes to the change of the ease for re-understanding the modified c . Since there is a “has-a” relationship between a whole-class and its every part-class, the ease of understanding the whole-class bases on the ease of understanding its part-classes. Thus, the change of understandability of every part-class will influence the understandability of the whole-class.

- For any $(c_1, c_2) \in AS_1 \setminus AS_0$,

$$\Delta U^a = \begin{cases} 0, & \exists (c'_1, c'_2) \in AS_0 \wedge (c'_1, c_1), (c'_2, c_2) \in GE_1 \\ \delta, & \text{otherwise} \end{cases} \quad (5)$$

where $a = (c_1, c_2)$ and δ is the impact factor of the association relationships to understandability.

In (5), for every increased association a , if there is also an association between the two classes whose subclasses are linked by a , then the modification does not change the understandability, that is, no extra effort must be made to understand a , otherwise we must make some effort to understand a . Intuitively, the modification that satisfies the precondition of (5) preserves the topological relation in the original UML metamodel (see Fig. 2(g)), thus it does not influence the understandability of the metamodel. Since a modification of an association does not influence the understandability of the two classes at the association ends compared to a modification of a composition that will influence the understandability of all whole-classes, the definitions of (4) and (5) are different.

Let U_0 be the understandability of the standard UML defined by M_0 and U_1 be the understandability of the extended UML defined by M_1 . There is

$$U_1 = U_0 + \Delta U, \quad (6)$$

where

$$\Delta U = U_1 - U_0 = \sum_{c \in C_1} \Delta U^c + \sum_{a \in A_1} \Delta U^a. \quad (7)$$

In the metric definition of the understandability deviation (i.e., (7)), it does not explicitly express the contribution of the generalizations and compositions to the change of understandability. The reason is that we have distributed their contribution to understandability into the computation of the understandability of classes (see (3) and (4)).

4.3 Empirical Validation

This subsection gives the empirical validation of the metric defined above. Since UML1.0 is not an available specification of OMG, we select UML1.1 as the standard UML, and UML1.4 [3], UML2.0 [4] and UML2.1 [2] as three extended UMLs. We mainly conducted our validation on use case diagram, profile [1] and state machine diagram in the three versions. Applying the proposed metric to them, we obtained the data shown in Table 1 by a metric tool we had developed. In [8], using three controlled experiments, authors calculated the correlation coefficients of the total number of attributes, the total number of generalizations and the total number of aggregations with the understandability of class diagram as 0.769, 0.679 and 0.52 respectively. Based on the result, we set the impact factors, by rounding the values given in [8], in our metric as $\alpha = 0.8$, $\beta = 0.7$, $\gamma = 0.5$ and $\delta = 0.5$ [2].

Controlled Experiments. For checking whether the metric results is consistent with our cognition in the real world, we carried out a set of controlled experiments to get the ease degree of understanding those measured diagrams. We randomly selected 30 undergraduate students in our school, who had taken courses about software engineering for one year and learnt object-oriented design with UML. Those students were divided into three groups and each group had 10

¹ This term firstly occurs in UML2.0, whose original is ExtensionMechanisms in UML1.1 and UML1.4. For simplicity, we do not distinguish the two names in this paper.

² In [8], authors had calculated two correlation coefficients of the total number of associations with the understandability of class diagram by the data in two experiments. According to the authors' analysis, the difference of the material in the two experiments resulted in the big difference of the two correlation coefficients (one is 0.3 and the other is 0.8). We get a mean of the two correlation coefficients as the impact factor of the associations (i.e., $\delta = 0.5$) in this paper.

Table 1. The understandability deviation of use case diagrams, profiles and state machine diagram in UML1.4, UML2.0 and UML2.1 compared with those in UML1.1 respectively ($\alpha = 0.8$, $\beta = 0.7$, $\gamma = 0.5$, $\delta = 0.5$)

UML	Use Case Diagram			Profile			State Machine		
	v1.4	v2.0	v2.1	v1.4	v2.0	v2.1	v1.4	v2.0	v2.1
ΔU	9.50	16.23	18.70	5.80	15.60	20.07	0.50	25.01	30.45

people. Firstly, the metamodel of use case diagram in UML1.1 was given to everyone in the three groups to understand and they must answer some questions in a questionnaire (see Appendix) that reflects whether they had understood the diagram. We used Understanding Time (UT) to record the time that the students spend on answering the questions. UT can be calculated by start time subtracting finish time. Secondly, we delivered the metamodels of use case diagrams in UML1.4, in UML2.0 and in UML2.1 to the first group, the second group and the third group respectively. At the same time, we required them to answer the corresponding questionnaires and recorded UTs. For profile and state machine diagram, we repeated above process, that is, the first step is to give the metamodels in UML1.1 to all students and calculate the UTs by questionnaires; the second step is to give metamodels in UML1.4, in UML2.0 and in UML2.1 to the first group, the second group and the third group respectively, and calculate the UTs by the corresponding questionnaires. Finally, we calculated the mean of UTs shown in Table 2 for every diagram in each UML version. With the UTs in UML1.1 as a base value, we calculated the difference values ΔUT (see Table 2) between UTs in UML1.4 (resp. UML2.0 and UML2.1) and the base value.

Note that there are many same or similar elements in the metamodels of different extension versions of UML. One person may understand a extension versions more easily after he had read another. To avoid this kind of intercrossing effect, we gave different versions to different groups, except the standard version (i.e., UML1.1) as a base.

Design of Questionnaire. All questions in these questionnaires were designed elaborately. Every one reflected a crucial difference between the self language version and other versions. For example, by comparison with the metamodel of UML1.1, a new class—Include—is added into the metamodel of UML1.4 by association relationships. It means that there could exist an include relationship between two use cases. In order to check whether a testee has perceived it, we ask a question—“Can an Include relationship exists between two use cases?”. Similarly, there is a new class—Extend—with a par-class Constraint (but Include without the part-class) in the metamodel of UML2.0. It means that we could give the condition that must hold for an extend relationship. In order to inspect whether a testee has perceived it, we ask a question—“Can we define a constraint for any relationship between two use cases?”. If a testee wants to answer these questions, regardless of right and wrong, he must try his best to comprehend the meaning of the metamodel of certain UML version. The longer the time that a

Table 2. Mean of Understanding Times and their difference values in the experiment on UML

UML		Use Case Diagram				Profile				State Machine			
		v1.1	v1.4	v2.0	v2.1	v1.1	v1.4	v2.0	v2.1	v1.1	v1.4	v2.0	v2.1
Mean of UT (sec.)	group I	120.0	169.0	—	—	143.0	179.0	—	—	301.0	355.0	—	—
	group II	110.0	—	294.0	—	133.0	—	338.0	—	323.0	—	515.0	—
	group III	116.0	—	—	319.0	137.0	—	—	364.0	289.0	—	—	587.0
	ΔUT (second)	—	49.0	184.0	203.0	—	36.0	205.0	227.0	—	54.0	192.0	298.0

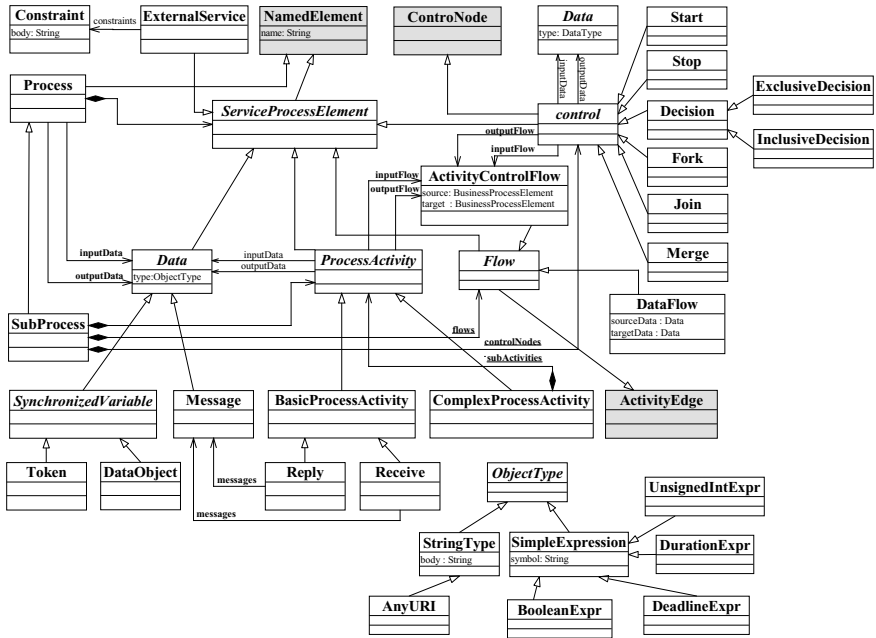
testee spends to answer these questions is, it means, the more difficult is it to understand the corresponding metamodel.

Additionally, we merely gave a small quantity of questions in one questionnaire, which was to avoid the negative mood of students, due to too many questions, to bother the objectivity of the experiment results.

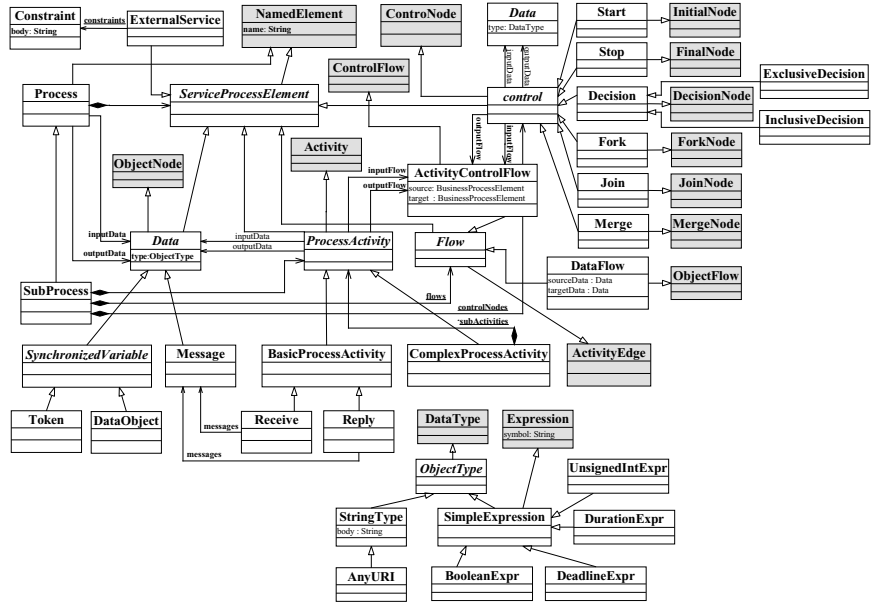
Result of Validation. According to the data in Table 1 and Table 2, we calculated the correlation coefficient between ΔU and ΔUT , $\rho = 0.91$, which indicates that a strong correlation exists between the metric values of the understandability deviation and the difference values of UT. We consider that UT reflects the ease degree of understanding an object. The bigger UT is, the worse is the understandability. Since the metric results is consistent with our cognition in the real world, the proposed metric is valid.

5 Evaluation of Extension Patterns for UML by Understandability Measurement

For modeling the work flow in a service-based project more effectively, our research group extended UML2.0 activity diagram [4]. Initially, we only considered the expression power of the desired modeling language and the ease of the extension. Under this consideration, we got the first version of the extended activity diagram called as Service Process Modeling Language version 1.0 (SPML1.0). The metamodel of SPML1.0 is shown in Fig. 4(a). However, in the end of the project, we wanted the modeling language to be used in future similar projects and decided to change the extension patterns to improve its understandability. As the result, we got the second version of SPML, i.e. SPML2.0, which had the same expression power as SPML1.0. The metamodel of SPML2.0 is shown in Fig. 4(b). Compared with SPML1.0, the main difference of the extension for SPML2.0 is that the increased elements in SPML2.0 metamodel tried their best to inherit the existing elements in activity diagram.



(a) metamodel of SPML1.0



(b) metamodel of SPML2.0

Fig. 4. The metamodels of SPML1.0 and SPML2.0. The grey is the original in the metamodel of UML activity diagram.

Based on the metamodel of UML2.0 activity diagram, we applied the proposed metric to the metamodels of SPML1.0 and SPML2.0 for evaluate their understandability, and the results is shown in Table 3. From the data in Table 3 we can find that the understandability deviation of SPML2.0 is less than that of SPML1.0. The metric result indicates that person will make $32.33 - 17.52 = 14.81$ unit effort to understand SPML2.0 less than to SPML1.0, under the precondition that he had understood UML2.0 activity diagram. Thus, we conclude that the understandability of SPML2.0 is better than that of SPML1.0. The reason of this conclusion is that unlike the metamodel of SPML1.0, many new classes in the metamodel of SPML2.0 inherit from some classes in the metamodel of UML2.0 activity diagram. Since users had been familiar with UML2.0 activity diagram, it is easier to understand those new elements in the metamodel of SPML2.0.

Table 3. The understandability deviation of SPML1.0 and SPML2.0 compared with UML2.0 activity diagram ($\alpha = 0.8$, $\beta = 0.7$, $\gamma = 0.5$, $\delta = 0.5$)

Version	SPML1.0	SPML2.0
ΔU	32.33	17.52

We repeated the controlled experiment mentioned in the subsection 4.3 on SPML. We divided the 30 students into two groups. Each group had 15 people. In the first step, we gave the metamodel of UML2.0 activity diagram to all students and got the UT by a questionnaire. In the second step, we gave the metamodel of SPML1.0 to the first group and the metamodel of SPML2.0 to the second group. By the corresponding questionnaires, we got the UTs of SPML1.0 and SPML2.0. In the last step, we calculated the mean of UTs for UML2.0 activity diagram, SPML1.0 and SPML2.0. All data are shown in Table 4. The metric results are consistent with the experiment data. The practice validates that the proposed metric is available.

Table 4. Mean of Understanding Times and their difference values in the experiment on SPML

Diagram		Activity Diagram	SPML1.0	SPML2.0
Mean of UT (second)	group I	500.7	1366.7	—
	group II	484.7	—	1008.7
ΔUT (second)		—	866.0	524.0

6 Conclusion

We study the measurement of understandability for modeling languages from the metamodel perspective. Based on measuring the deviation of understandability between extended UMLs and the standard UML, we propose the metric model of understandability for extended UMLs. This metric can be used to evaluate

different extended UMLs with the same modeling power on the understandability. Furthermore, the proposed metric can guide users to select an extension patterns of UML with better quality to achieve their goals. The proposed metric is empirically validated by different UML versions. We also give an example in a real project to explain the application of the proposed metric. Our work was conducted on UML, but the method can be applied to other modeling languages. Currently, we only consider the influence on the understandability of a modeling language resulting from the complexity of its syntax, rather than its semantic. In addition, the sample space of our controlled experiments is not big enough. These are the limitations of this study. In the future, other quality attributes of modeling languages, such as maintainability and expressive power, will be studied from metamodel perspective.

Acknowledgments. The authors would like to thank the anonymous referees for their helpful comments on this paper. This work is supported by the National Natural Science Foundation of China (No.61003025) and the Jiangsu Province Research Foundation (BK2010170).

References

1. Object Management Group: UML semantics version 1.1. Document ad/97-08-04, OMG (1997)
2. Object Management Group: Unified Model Language (UML): Superstructure version 2.1.2. Document formal/07-11-02, OMG (2007)
3. Object Management Group: OMG Unified Model Language Specification Version 1.4. Document formal/01-09-67, OMG (2001)
4. Object Management Group: Unified Model Language (UML): Superstructure version 2.0. Document formal/05-07-04, OMG (2005)
5. Zhang, Y., Liu, Y., Zhang, L., Ma, Z., Mei, H.: Modeling and checking for non-functional attributes in extended UML class diagram. In: Proceedings of 32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC 2008), pp. 100–107. IEEE Computer Society, Los Alamitos (2008)
6. Wada, H., Suzuki, J., Oba, K.: Modeling non-functional aspects in service oriented architecture. In: Proceedings of IEEE International Conference on Services Computing (SCC 2006), pp. 222–229. IEEE Computer Society, Los Alamitos (2006)
7. Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., Yu, E.: Evaluating goal models within the goal-oriented requirement language. *International Journal of Intelligent Systems* 25(8), 841–877 (2009)
8. Genero, M., Poels, G., Manso, E., Piattini, M.: Defining and validating metrics for UML class diagrams. In: Genero, M., Piattini, M., Calero, C. (eds.) *Metrics for Software Conceptual Models*, pp. 99–159. Imperial College Press (2005)
9. Genero, M., Piattini, M., Manso, E.: Finding “early” indicators of UML class diagrams understandability and modifiability. In: Proceedings of the 2004 International Symposium on Empirical Software Engineering (ISESE 2004), pp. 207–216. IEEE Computer Society, Los Alamitos (2004)
10. Genero, M., Piattini, M., Manso, E., Cantone, G.: Building UML class diagram maintainability prediction models based on early metrics. In: Proceedings of the Ninth International Software Metrics Symposium (METRICS 2003), pp. 263–275. IEEE Computer Society, Los Alamitos (2003)

11. Bansiya, J., Davis, C.G.: A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on Software Engineering* 28(1), 4–17 (2002)
12. Object Management Group: Meta Object Facility (MOF) specification version 1.3. Document formal/00-04-03, OMG (2000)
13. Jiang, Y., Shao, W., Zhang, L., Ma, Z., Meng, X., Ma, H.: On the Classification of UML's Meta Model Extension Mechanism. In: Baar, T., Strohmeier, A., Moreira, A., Mellor, S.J. (eds.) *UML 2004*. LNCS, vol. 3273, pp. 54–68. Springer, Heidelberg (2004)
14. Ma, H., Shao, W., Zhang, L., Ma, Z., Jiang, Y.: Applying OO Metrics to Assess UML Meta-models. In: Baar, T., Strohmeier, A., Moreira, A., Mellor, S.J. (eds.) *UML 2004*. LNCS, vol. 3273, pp. 12–26. Springer, Heidelberg (2004)
15. Object Management Group: Meta Object Facility (MOF) core specification version 2.0. OMG Available Specification formal/06-01-01, OMG (2006)
16. Briand, L.C., Wüst, J.: Modeling development effort in object-oriented systems using design properties. *IEEE Transactions on Software Engineering* 27(11), 963–986 (2001)

Appendix: The Questionnaires for UML Use Case Diagram

For the limit of space, we only give the questionnaires of use case diagram in UML1.1, UML1.4, UML2.0 and UML2.1 respectively.

Questionnaire for UML1.1 Use Case Diagram

Diagram: The metamodel of UML1.1 use case diagram (Omitted)

START TIME: _____

With the metamodel shown above, answer the following questions:

- Does UseCase have attributes and operations?
- Is there a relationship between UseCase and Actor?
- Can an Extend relationship exist between two use cases?
- Can a use case have extension point?

FINISH TIME: _____

Questionnaire for UML1.4 Use Case Diagram

Diagram: The metamodel of UML1.4 use case diagram (Omitted)

START TIME: _____

With the metamodel shown above, answer the following questions:

- Can an Extend relationship exists between two use cases?
- Can an Include relationship exists between two use cases?
- Is there a way in which we can describe in a use case where it may be extended?
- Is UseCaseInstance an instance of UseCase?

FINISH TIME: _____

Questionnaire for UML2.0 Use Case Diagram

Diagram: The metamodel of UML2.0 use case diagram (Omitted)

START TIME: _____

With the metamodel shown above, answer the following questions:

- Can an Include relationship exists between two use cases?
- Can we name the relationships between two use cases?
- Does UseCase inherit from BehavioredClassifier or just Classifier?
- Can we define a constraint for any relationship between two use cases?

FINISH TIME: _____

Questionnaire for UML2.1 Use Case Diagram

Diagram: The metamodel of UML2.1 use case diagram (Omitted)

START TIME: _____

With the metamodel shown above, answer the following questions:

- Can an Include relationship exists between two use cases?
- Can we name the relationships between two use cases?
- Does Actor inherit from BehavioredClassifier or just Classifier?
- Can we define a constraint for any relationship between two use cases?

FINISH TIME: _____

Dealing with Dependencies among Functional and Non-functional Requirements for Impact Analysis in Web Engineering

José Alfonso Aguilar¹, Irene Garrigós²,
Jose-Norberto Mazón², and Anibal Zaldívar¹

¹ Señales y Sistemas (SESI)
Facultad de Informática Mazatlán
Universidad Autónoma de Sinaloa, México
{ja.aguilar, azaldivar}@maz.uasnet.mx

² Department of Software and Computing Systems (DLSI)
University of Alicante, Spain
{igarrigos, jnmazon}@dlsi.ua.es
<http://sisis.maz.uasnet.mx>

Abstract. Due to the dynamic nature of the Web as well as its heterogeneous audience, web applications are more likely to rapidly evolve leading to inconsistencies among requirements during the development process. With the purpose to deal with these inconsistencies, web developers need to know dependencies among requirements considering that the understanding of these dependencies helps in better managing and maintaining web applications. In this paper, an algorithm has been defined and implemented in order to analyze dependencies among functional and non-functional requirements (in a goal-oriented approach) for understanding which is the impact derived from a change during the Model-Driven Web Engineering process. This Impact Analysis would support web developer in selecting requirements to be implemented ensuring that web applications finally satisfy the audience.

Keywords: Impact Analysis, Goal-Oriented Requirements Engineering, Web Engineering, Model-Driven Web Engineering.

1 Introduction

Requirements in web engineering (WE) tend to rapidly evolve due to the dynamic nature of the Web [1]. This continuous evolution may lead to inconsistencies among requirements and the web application. This discrepancy may hinder the web developers to understand how the changes in the requirements affects the final web application. In this context, a crucial issue in WE is the Impact Analysis (IA) of requirements, which is the task of identifying the potential consequences of a change or estimating what needs to be modified to accomplish a change [2]. We define a “change” as any modification on a web requirement, i.e., add or delete a requirement. Usually, IA has been done intuitively by web applications

developers after some cursory examination of the code and documentation. This may be sufficient for small web applications, but it is not enough for sophisticated ones (i.e. a money exchange and transfer web application). In addition, empirical investigation shows that even experienced web applications developers predict incomplete sets of change impacts [3].

In order to effectively analyze the impact of requirements in web applications the dependencies among requirements should be explicitly considered to better manage changes. Actually, inconsistencies are defined as negative dependencies among the set of requirements, caused by the fact that requirements often originate from stakeholders with different or conflicting viewpoints. Commonly, the IA is performed only on functional requirements (FRs), leaving aside the non-functional requirements (NFRs) as shown in [4,5]. In software engineering (SE), FR describes system services, behavior or functions, whereas NFR, or quality requirements, specify a constraint on the system or on the development process [6]. According to [7], we believe that the NFRs must be considered from the beginning of the development process and that the IA should be done on both kinds of requirements: FR and NFRs.

Interestingly, the inclusion of goal-oriented requirements engineering (GORE) in WE [8,9,10,11] offers a better analysis in web application design due to the fact that requirements are explicitly specified in goal-oriented models, thus supporting developers in evaluating the implementation of certain requirements (FR and NFRs) for designing successful software. In the GORE field, FR are related to goals and sub-goals whereas NFRs are named softgoals, commonly used to represent objectives that miss clear-cut criteria. Specifically, finding right tradeoffs for dependent NFRs is an important step when impact of requirements is being analyzed [12], i.e., a web application without passwords is usable, but not very secure, increased usability reduce security or increased security reduce usability. Unfortunately, finding these tradeoffs among dependent web requirements is not a trivial task due to the dynamic nature of the Web. Therefore, it is important to improve the use of the requirements engineering (RE) in further steps of the web application development process.

There have been many attempts to provide techniques and methods to deal with some aspects of the RE process for the development of web applications, but there is still a need for solutions which enable the designer to know which requirements are affected because of a change in the web application design besides of considering the NFRs involved in the development process.

The proposal presented through the paper consist in a GORE approach for supporting web developers in analyzing the impact of a change by considering both FR and NFRs. To this aim, an algorithm has been defined to support web developer in (i) clearly identifying dependencies among FR and NFRs, and (ii) automatically performing the IA by determining right tradeoffs. The main benefit of our approach is that provides information about the different design alternatives, thus allowing developers to make more informed design decisions for implementing a web application that fully-satisfies FR, while there is a tradeoff among NFRs.

This paper is an extension of our recent work [13] about the importance of considering IA in our GORE approach. In particular, the novelty of our ongoing work presented in this paper consists of: (i) the implementation of the UML-Profile to adapt the i^* framework in the Web domain as a metamodel, (ii) the development of a prototype tool for the web requirements specification as a proof of concept of our approach, (iii) the implementation of model-to-model transformation rules (with a high degree of automation) to derive the web application conceptual models, and (iv) the implementation of the algorithm for IA in goal-oriented models.

The remainder of this paper is structured as follows: Section 2 presents some related work relevant to the context of this work. Section 3 describes the GORE proposal where is found the contribution of this work and introduces a running example for demonstration purposes. The algorithm for IA in goal-oriented models and its application is described in Section 4. In Section 5 is presented the current implementation of this approach. Finally, the conclusion and future work is presented in Section 6.

2 Related Work

In our previous work [14], a systematic literature review has been conducted for studying requirement engineering techniques in the development of web applications. Our findings showed that most of the web engineering approaches focus on the analysis and design phases and do not give a comprehensive support to the requirements phase (such as OOHDM [15], WSDM [16] or Hera [17]). Additionally, we can also conclude that the most used requirement analysis technique is the UML (Unified Modeling Language) use cases. This technique has proved to be successful in the common software development process to deal with the requirements specification in both textual and diagram form. But unfortunately, this technique is not enough to deal with aspects such as navigation in the web application development process even though is applied by some of the most remarkable web engineering approaches such as OOWS [18], WebML [19], NDT [20] and UWE [21].

In addition, none of the aforementioned WE approaches perform the analysis and modeling of the users' needs for ensuring that the web application satisfies real goals, i.e. users are not overloaded with useless functionalities, while important functionalities are not missed. We believe that these facts are an important issue that limits a broader use of these approaches. In this sense, to the best of our knowledge, the only approaches that use GORE techniques for WE have been presented in [22,23]. Unfortunately, although these approaches use the i^* modeling framework [24,25] to represent requirements in web domain, they do not benefit from every i^* feature because don't use all the expressiveness of the i^* framework to represent the special type of requirements of the web applications such as the related with navigational issues. To overcome this situation, our previous work [10] adapts the well-known taxonomy of web requirements presented in [26] for the i^* framework.

Regarding approaches that consider NFRs requirements from early stages of the development process, in [27] the authors propose a metamodel for representing usability requirements for web applications and in [7] the authors present the state-of-the-art for NFRs in model-driven development (MDD), as well as an approach for considering NFRs into a MDD process from the very beginning of the development process. Unfortunately, these works overlook how to analyze and evaluate the impact among FRs and NFRs. However, some interesting works have been done in this area [28] and [29]. These works evaluate i^* models based upon an analysis question (what-if) and the human judgment. To this aim, this procedure uses a set of evaluation labels that represent the satisfaction or denial level of each element in the i^* model. First of all, initial evaluation labels reflecting an analysis question are placed in the model. These labels are then propagated throughout the model by using a combination of set propagation rules and the human judgment. The results of this propagation are interpreted in order to answer the stated question. Unfortunately, these general approaches have not been adapted to web engineering.

The motivation regarding this proposal relies in the fact that the works previously mentioned are focused on how to analyze i^* models (goal-oriented models) to answer a particular question (what-if) without considering the goal satisfaction (the organizational objectives). Thus, our proposal is focused on how to evaluate the impact derived from a change in the i^* requirements model in order to use it to offer to the designer a better form to analyze design options from the web application. For this purpose, the requirements are classified according the taxonomy of web requirements defined in [26] in the i^* requirements model. In addition, our proposal brings out alternative paths to satisfy the goals bearing in mind the softgoals tradeoff considering the softgoals from the beginning of the web application development process.

3 Goal-Oriented Requirements Analysis in Web Engineering

This section describes our proposal to specify requirements in the context of the A-OOH (Adaptive Object-Oriented Hypermedia) Web modeling method [30] by using goal-oriented models [10]. A-OOH is an extension of the OOH (Object-Oriented Hypermedia) [31] method with the inclusion of personalization strategies. The development process of this method is founded in the MDA (Model-Driven Architecture). MDA is an OMG's standard and consists of a three-tier architecture with which the requirements are specified at the Computational Independent Model (CIM), from there are derived the Web application conceptual models which corresponds with the Platform Independent Model (PIM) of the MDA. Finally, the Web application conceptual models are used to generate the implementation code; this stage corresponds with the Platform Specific Model (PSM) from the MDA standard. A crucial part of MDA is the concept of transformation between models either model-to-model (M2M) or model-to-text (M2T). With the M2M transformations is possible the transformation from

a model in other one. To use the advantages of MDA, our proposal supports the automatic derivation of Web conceptual models from a requirements model by means of a set of M2M transformation rules defined in [10,32].

We shortly describe next an excerpt of the i^* framework which is relevant for the present work. For a further explanation, we refer the reader to [24,25]. The i^* framework consists of two models: the strategic dependency (SD) model to describe the dependency relationships (represented as \dashv) among various actors in an organizational context, and the strategic rationale (SR) model, used to describe actor interests and concerns and how they might be addressed. The SR model (represented as \odot) provides a detailed way of modeling internal intentional elements and relationships of each actor (\odot). Intentional elements are goals (\odot), tasks (\diamond), resources (\square) and softgoals (∞). Intentional relationships are means-end links (\dashv) representing alternative ways for fulfilling goals; task-decomposition links (\dashv) representing the necessary elements for a task to be performed; or contribution links ($\xrightarrow[\text{next}]{\text{help}}$) in order to model how an intentional element contributes to the satisfaction or fulfillment of a softgoal.

Even though i^* provides good mechanisms to model actors and relationships between them, it needs to be adapted to the web engineering domain to reflect special web requirements that are not taken into account in traditional requirement analysis approaches. As the A-OOH approach is UML-compliant, we have used the extension mechanisms of UML in order to adapt the i^* modeling framework to the taxonomy of Web requirements (Content, Service, Navigational, Layout, Personalization and Non-Functional Requirements) presented in [26]. To do so, (i) we defined a profile to formally represent the adaptation of each one of the i^* elements with each requirement type from the Web requirements clasification adopted [11]; and (ii) we implemented this profile in an EMF (Eclipse Modeling Framework) metamodel adding new EMF clases according to the different kind of Web requirements: the Navigational, Service, Personalization and Layout requirements extends the Task element and the Content requirement extends the Resource class. It is worth noting that NFRs can be modeled by directly using the softgoal element. In Figure 1 can be seen an extract of the EMF metamodel for Web requirements specification using the i^* framework. The metamodel has been implemented in the Eclipse [33] IDE (Integrated Development Enviroment).

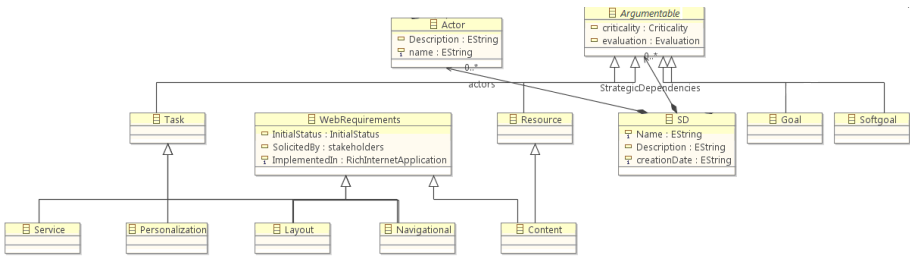


Fig. 1. An overview of the i^* metamodel implemented in Eclipse (EMF)

3.1 A Practical Case: A Web Requirements Specification

Based on the i^* modeling framework adaptation for the web domain, next is described a web requirements specification for the Conference Management System (CMS) which will be used through the paper as a running example. The purpose of the system is to support the process of submission, evaluation and selection of papers for a conference [34]¹. The requirements specification is shown in Figure 2 which models a part of the CMS focused in the process of selecting the review process. Four actors participate in the CMS, but due to space limitations, for this example only the author, reviewer and system (named “*WebApplication*”) actors were considered. It is important to highlight that each element from Figure 2 corresponds to a requirements type from the taxonomy previously mentioned, i.e., the content requirement (Content) from the taxonomy is displayed with the notation “*Resource*” from i^* and the navigational (Navigational) and service (Service) requirements with the symbol “*Task*” from i^* , both with their respective associations (*decomposition-links*). A decomposition-link between two elements means that a requirement (“*Task*”) is decomposed in one or more sub-requirements (“*Sub-Tasks*”), Figure 2 depicts a correct scenario of the requirement decomposition by introducing the navigational requirement “*Blind Review Process*”, decomposed in two sub-requirements named “*Download papers without authors’ name*” (Service) and “*Review Paper*” (Navigational). Also, the labels (✓) and (✗) are used to represent the requirements currently implemented in the Web application.

Three actors are detected that depend on each other, namely “*Reviewer*”, “*Author*” and “*Conference Management System*” (named as “*WebApplication*” in Figure 2). The reviewer needs to use the CMS to “*Review paper*”. The author depends on the CMS in order to “*Paper be reviewed*”. These dependencies and the CMS actor are modeled by a SD and SR models in Figure 2. The goal of the CMS actor is “*Process of review of papers be selected*”. To fulfill this goal, the SR model specifies that one of the two navigational requirements: “*Blind review process*” or “*Normal review process*” should be performed. In this running example, the path to achieve the goal of the CMS actor is by means of the navigational requirement “*Blind review process*”, all the requirements implemented for this path are labeled with (✓). We can observe in the SR model that some navigational and service requirements are decomposed in other requirements, some of them affects positively or negatively some non-functional requirements, i.e., the service requirement “*Download paper without authors’ name*” needs the content requirement “*Papers*”, also, affects positively the softgoal “*Privacy be maximized*” and in some negatively form the softgoal “*Obtain more complete info*”. This fact is very important to see how to satisfy the goal “*Process of review of papers be selected*” considering the web application softgoals. One viable solution to this issue is maximizing or minimizing the contribution from requirements to softgoals in order to find a path to fully-satisfy the goal.

¹ The complete specification of the case study can be found at:

<http://users.dsic.upv.es/~west/iwost01>

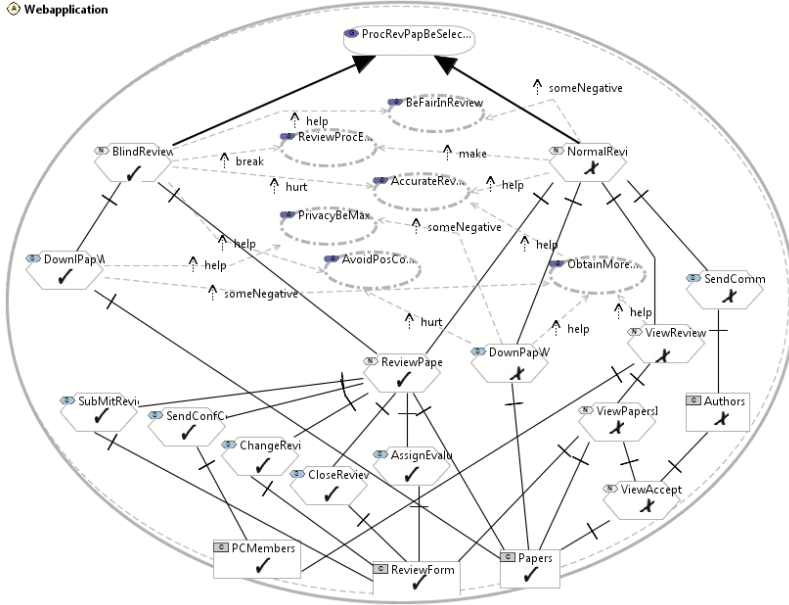


Fig. 2. Part of the Conference Management System (CMS) requirements expressed in a SR and SD (i^*) models

4 An Impact Analysis Algorithm for Goal-Oriented Requirements in Web Engineering

In this section is briefly introduced our algorithm for impact analysis in goal-oriented models [13]. The algorithm provides a way to analyze the impact of a change in an A-OOH requirements model. With the algorithm, the Web developer will be able to evaluate the effect of the change and select the best among several design options to fully satisfy goals based on maximizing softgoals (it is worth noting that, in this paper, our focus is the implementation of the i^* metamodel and the development of a prototype tool for the Web requirements specification besides the implementation of the algorithm for impact analysis presented in [13]).

The algorithm is designed to be applied in i^* requirements model considering each type of contributions made by the intentional elements to the softgoals (“Help”, “Hurt”, “Some +”, “Some -”, “Some +”, “Break” and “Make”). Considering the contributions made by the intentional elements (task element from i^* model) to the softgoals, the algorithm evaluates the impact in the requirements model resulting from removing any element of the A-OOH conceptual models. In this way, it is determined which new requirements should be included in the A-OOH conceptual models for maximizing softgoal satisfaction although

some requirements have been removed. To do this, the designer must have to find tradeoffs between the softgoals.

4.1 Performing the Impact Analysis

For the sake of understandability, the following scenario is assumed along this section: the Web developer decides deleting from the domain model of A-OOH the elements that correspond to the requirement “*Download papers without authors’ name*”. It is necessary to know which other requirements are affected by this change. In addition, this action implies that the goal “*Process of review of papers be selected*” can not be satisfied. Thus, it is necessary to search for alternative paths in the i^* requirements model (if there any) in order to fully-satisfy the goal “*Process of review papers be selected*”. To this aim, our algorithm is triggered. The execution of the impact analysis algorithm is detailed next.

The first step to execute our algorithm consists of applying a set of preconditions (explained in detail in our previous work [13]). For this running example, the preconditions result true, it means that there is any problem to the algorithm has been executed.

Next, it is necessary to develop a list of the requirements (implemented or not) that contribute to any softgoal in the i^* requirements model (see Table 1). Also, if a softgoal contributes to other one, the softgoal must be added to the list too.

Table 1. The requirements contributions to softgoals

Requirements	“S1”	“S2”	“S3”	“S4”	“S5”	“S6”
<i>Blind Review Process</i>	Help	Break	Hurt	Help	-	-
<i>Download Papers Without Authors’ Name</i>	-	-	-	-	Help	Some -
<i>Normal Review Process</i>	Some -	Make	Help	-	-	-
<i>Download Paper With AuthorsName</i>	-	-	-	Hurt	Some -	Help
<i>View Review Process Status</i>	-	-	-	-	-	Help
<i>Obtain More Complete Info</i>	-	-	Help	-	-	-

Table 1 highlights in bold the requirement to be removed (“*Download papers without authors’ name*”). This table shows a requirements list (FRs and NFRs) and their type of contributions to the softgoals where S1 corresponds to softgoal “*Be fair in review*” from requirements model, S2 to “*Review process easier*”, S3 represents “*Accurate review process*”, S4 conforms to “*Avoid possible conflicts of interest*”, S5 its the “*Privacy be maximized*” softgoal and S6 refers to “*Obtain more complete info*”.

The next step is to identify the number of softgoals affected by the requirement to be removed. If necessary, a list of the softgoals that receive a contribution from the requirement to be removed is made. In this example, the requirement to be removed is “*Download papers without authors’ name*”, this one affects two softgoals: “*Privacy be maximized*” and “*Obtain more complete info*” S5 y S6 respectively (see Table 1).

For each softgoal that receives a contribution from the requirement to be removed, we search for a non-implemented requirement of which contribution compensates the possible elimination of the requirement to be removed. To do this, it is necessary to apply a set of heuristics defined in [13]. For example, the softgoal “*Privacy be maximized*”, receives a *Help* from the requirement to be removed, thus being necessary searching for a non-implemented requirement to contribute to this softgoal. In this case, only the requirement “*Download papers with authors’ name*” contributes negatively to this softgoal (Some -), for this reason, applying the heuristics described in [13], specifically the heuristic H2², the requirement “*Download papers with authors’ name*” could be implemented.

Considering the softgoal “*Obtain more complete info*”, it receives a *Some -* contribution from the requirement to be removed, thus being necessary searching for a non-implemented requirement to contribute to this softgoal. In this case, two requirements (positively) contribute to this softgoal, “*Download papers with authors’ name*” and “*View review process status*” with a *Help* contribution link, hence, the heuristic H3³ applies for this softgoal, thus, these requirements should be implemented.

After analyzing the softgoals contributions, the next step is searching for any softgoal in the requirements list that contributes to another softgoal. In this example, the softgoal “*Obtain more complete info*” makes a *Help* contribution to the softgoal “*Accurate review process*”, thus, the next step consists of searching for the requirement that makes a contribution to the softgoal and applying the heuristics. The requirement that makes a contribution to the softgoal “*Accurate review process*” is “*Normal review process*”, this contribution is *Help* (see Figure 2), hence, according to H3 this requirement must be implemented.

After these steps, Table 2 shows requirements that could be implemented to fully-satisfy the goal “*Process of review papers be selected*” after having removed the requirement “*Download papers without authors’ name*”. Next, it is necessary to evaluate the heuristics assigned to each requirement to know what could be implemented.

Table 2. Non-implemented requirements that contributes to softgoals

Intentional element	“S5”	“S4”	“S6”	“S3”	Result
<i>Download Papers With Authors’ Name</i>	H2 (Some -)	H1 (Hurt)	H3 (Help)	-	Implement
<i>Normal Review Process</i>	-	-	-	H3 (Help)	Implement
<i>View Review Process Status</i>	-	-	H3 (Help)	-	Implement

Table 2 shows the results after having performed the algorithm. In this table the requirements that must be implemented in order to fully-satisfy the goal

² H2: if the contribution of the requirement to remove is *help* than the contribution of the requirement to implement, but the contribution to be implemented is the *Some -* type, the requirement **could** be implemented.

³ H3: if the contribution of the requirement to remove is *Hurt* or *Some -*, and the contribution of the requirement to implement is *Help* or *Some +*, the requirement **should be** implemented.

“Process of review papers be selected” are shown. To do this, it is necessary to evaluate the contribution type of each requirement, i.e., the navigational requirement “Download papers with authors’ name” negatively contributes (Some -) to the softgoal “Privacy be maximized”, thus hurting the softgoal “Avoid possible conflicts of interest” and helping the softgoal “Obtain more complete info”. Therefore, by using the human judgment the navigational requirement “Download papers with authors’ name” can be implemented. For the navigational requirement “Normal review process”, it is easier to determine whether it can be implemented because it only contributes to one softgoal, the “Accurate review process”, hence its contribution is Help, this requirement must be implemented. Finally, the navigational requirement “View review process status” positively contributes to the softgoal “Obtain more complete info”, consequently this requirement must be implemented.

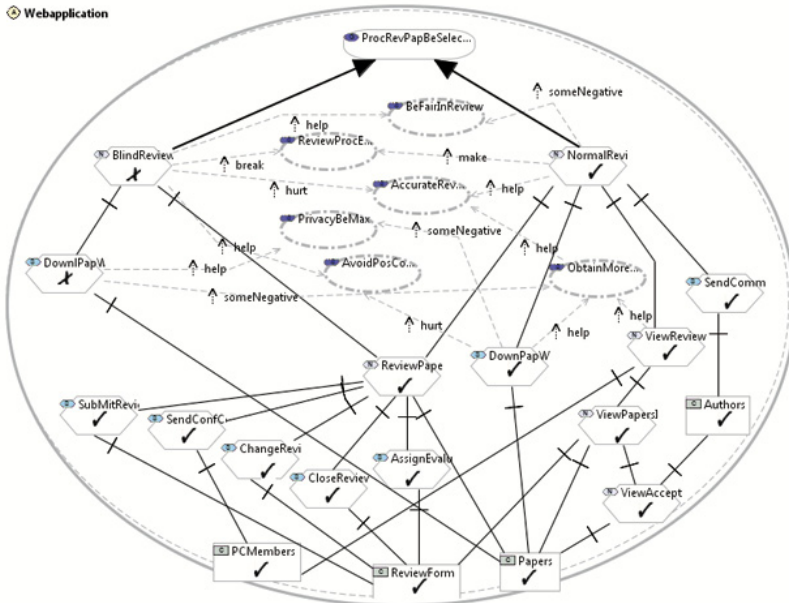


Fig. 3. Conference Management System requirements expressed in a SR and SD Models with the alternative path to fully satisfy the goal “Process of review papers be selected”

The final step is to apply a postcondition, in this running example it is necessary to implement the navigational requirements “View papers info” and “View Accepted/Rejected papers” because these requirements are associated with the navigational requirement “View Review Process Status”. In addition, the content requirement “Authors” and the service requirement “Send Comments to Authors” must be implemented too in order to implement the alternative path to fully satisfy the goal “Process of review papers be selected”. Hence, the content requirement “Authors” is associated with the navigational requirement “View

Accepted/Rejected papers” and the service requirement *“Send Comments to Authors”* is related with the navigational requirement *“Normal review process”*.

After finishing the execution of the algorithm, we obtain the requirements that are directly and indirectly affected by the deletion of the requirement *“Download papers without authors’ name”*. Moreover, the algorithm can find out which requirements must be implemented to continue satisfying the goal considering the contributions received from the softgoals. In this running example the requirements to implement are: *“Download papers with authors’ name”*, *“Normal review process”* and *“View review process status”*. Finally, according to the post-condition the requirements *“View papers info”*, *“View Accepted/Rejected papers”*, *“Authors”* and *“Send Comments to Authors”* must be implemented too. Figure 3 shows the final requirements model with the alternative path implemented to fully-satisfy the goal *“Process of review papers be selected”*.

5 Open Source Implementation Framework

In this section we describe in detail the implementation of the impact analysis algorithm within our approach for goal-oriented requirements analysis in Web engineering. To this aim, we have combined a set of technologies such as Eclipse, EMF (Eclipse Modeling Framework), GMF (Graphical Modeling Framework) within the GMP (Graphical Modeling Project) and Java.

Eclipse is an open source IDE used as a software platform to create integrated development environments; within Eclipse, the EMF project is a modeling framework and code generation facility for building tools and other applications based on a structured data model (abstract syntax). Also, the facilities for creating metamodels and models are provided by the metamodel Ecocore; by using the facilities offered by EMF, it is possible to create a visual representation of the elements defined within the EMF metamodel by means of GMP (concrete syntax). The Eclipse Graphical Modeling Project (GMP) provides a set of generative components and runtime infrastructures for developing graphical editors based on EMF and GEF (Graphical Editing Framework). Both the Eclipse Modeling Framework (EMF) and the Graphical Modeling Framework (GMF) are capable of generating editor plug-ins. Next, each one of the steps performed for the implementation framework is described.

The first step is the implementation of the web requirements metamodel. The requirements metamodel was created using the EMF metamodel to incorporate a number of taxonomic features for the specification of Web requirements. With the implementation of this metamodel has been possible to adapt the i^* modeling framework in the Web domain, with which is possible to model the needs and expectations of the stakeholders of the Web application. The classification of Web requirements presented in Section 3 have been incorporated as Ecocore classes to represent each type of the requirements classification.

Once the metamodel has been implemented it is necessary to provide a graphical tool to assist the designer with the requirements specification. To do so, we have implemented a graphical editor using the GMF technology from the Eclipse Graphical Modeling Project.

In Figure 4, it is displayed a screenshot of the graphical editor implemented (called WebRED) by combining the Web requirements metamodel and the GMF technology. In the center of the figure is described the requirements specification for the the Conference Management System from the case study presented in the Section 3.

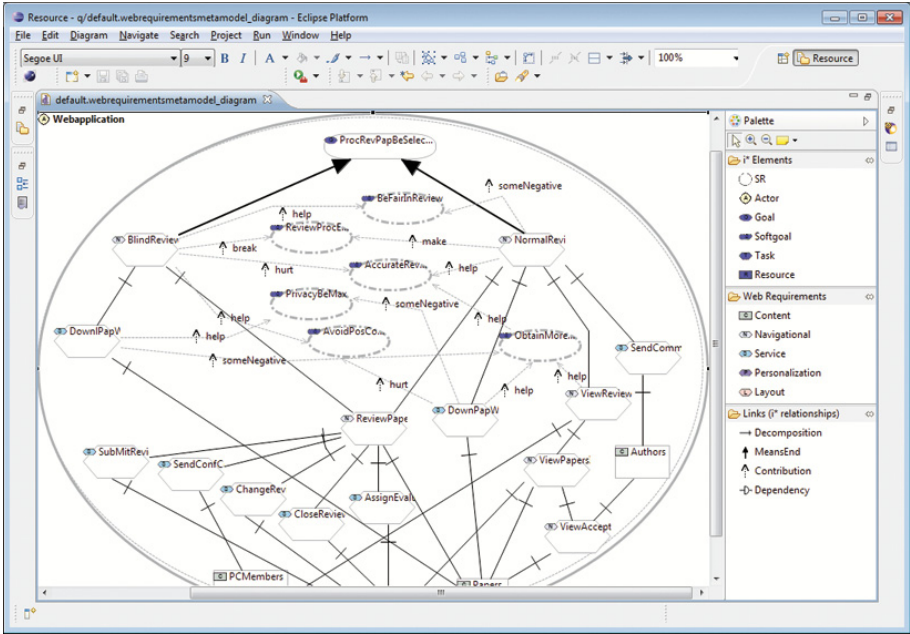


Fig. 4. WebRED, Web Requirements Editor

As the reader can see, we have implemented each one of the elements of the web requirements metamodel in the tool, thus the designer can model each requirement type from the clasification adopted (described in the Section 3). Also, WebRED provides the basic elements to model clasical goal-oriented models such as Task, Resource, Goal and Softgoal. The palette for drawing the different elements of the i^* models for requirements specification can be seen on the right side of the Figure 4. At the top of the figure, there are those elements required to specify goal-oriented models according to the i^* notation. Also, the requirements clasification adopted to specify requirements in the Web domain are in the right-center of the figure. Finally, the different types of relationships used in the i^* modeling framework, with which the elements can be associated, are the right-botton of the figure.

With regard to the impact analysis support, this is performed in an automatic manner. When the designer specifies the requirements of the Web application by

⁴ <http://code.google.com/p/webred/>

using the WebRED editor it is possible to know which requirements are affected due to a change in the Web application conceptual models. The impact of a change in the requirements can be consulted by the designer by means of a screen (window) and by means of a PDF (Portable Document File) report.

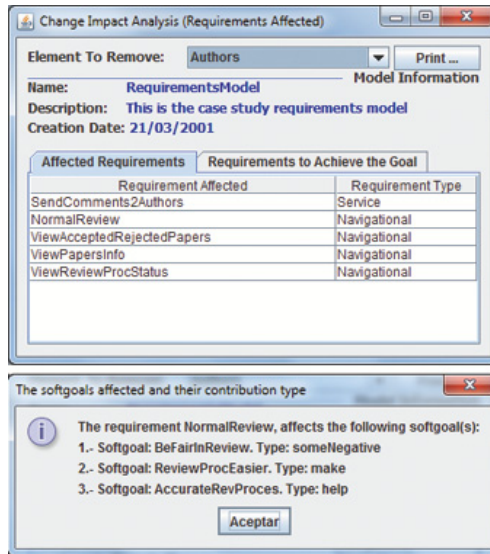


Fig. 5. Screenshot of the impact analysis support offered by the Web Requirements Editor (WebRED)

In Figure 5, it is shown a screenshot of the impact analysis support offered by the WebRED editor. At the top of the figure, the main window for the impact analysis option is displayed. At the top we find the name of the element affected by a change originated in any of the Web application conceptual models described as “Element to remove”. Also the information about the model currently selected is shown in the main window. In this particular case, this information is about the requirements model, thus including the name, description and creation date of the requirements model. Next, there is a tabbed pane with two options for the designer. The first one shows a list of the requirements affected if and only if the requirement “Authors” is removed; also, is showed the type of each one of the affected requirements. Moreover, if the designer selects one of the requirements listed by double clicking on it, a new window (message dialog) is showed with a list of the softgoals affected by the requirement selected from the list (Figure 5, down). This softgoal’s list shows the strength of the contribution made by the requirement affected to the softgoals. On the other hand, in the second tab, the requirements to be implemented (only when it is necessary) by the designer to still continue satisfying the goal are listed.

To conclude, the main window for the impact analysis option in the WebRED editor allows the designer to print a report in PDF (Portable Document File) format with which the designer can check the affected requirements.

6 Conclusions and Future Work

Owing the dynamic idiosyncrasy of the Web, its heterogeneous audience and its fast evolution, web applications should consider a RE process as complete as possible with the purpose of manage the requirements reflecting specific needs, goals, interests and preferences of each user or users types, besides to consider the NFRs from the early phases of the development process since these are not considered with sufficient importance [7].

To sum up, in this work we have presented: (i) a methodology based on the i^* modelling framework to specify web requirements considering the NFRs (softgoals) so that it allows the designer to make decisions from the beginning of the development process that would affect the structure of the envision web application in order to satisfy users needs; (ii) an algorithm to analyze the impact derived from a change done in the requirements model and the ability to find an alternative path to fully-satisfy the goal by means of the softgoals tradeoff; and (iii) an open source prototype tool that assists the web application designer performing the requirements specification and the impact analysis derived from a change in requirements. In this form, the designer can improve the quality of the requirements model analyzing the balance of the softgoals with the stakeholders.

Our short-term future work includes the definition of a metamodel to help to record the relationships among FR and NFRs improving the impact analysis report. Besides, it is important to remark that the graphical editor is the basis for a prototype tool for the development of Web applications using the MDA paradigm.

Finally, note that this work has been done in the context of the A-OOH modeling method; however it can be applied to any web modeling approach.

Acknowledgments. This work has been partially supported by: Universidad Autónoma de Sinaloa (Mexico), Consejo Nacional de Ciencia y Tecnología Mexico (CONACYT), Programa Integral de Fortalecimiento Institucional (PIFI 2011), Programa de Mejoramiento del Profesorado (PROMEP) at Secretaría de Educación Pública, Mexico and MANTRA (GRE09-17) from the University of Alicante.

References

1. Ginige, A.: Web engineering: managing the complexity of web systems development. In: SEKE, pp. 721–729 (2002)
2. Arnold, R., Bohner, S.: Impact analysis-towards a framework for comparison. In: Conference on Software Maintenance (CSM-93), pp. 292–301. IEEE (2002)
3. Lindvall, M., Sandahl, K.: How well do experienced software developers predict software change? Journal of Systems and Software 43(1), 19–27 (1998)
4. Zhang, S., Gu, Z., Lin, Y., Zhao, J.: Celadon: A change impact analysis tool for Aspect-Oriented programs. In: 30th International Conference on Software Engineering (ICSE), pp. 913–914. ACM (2008)

5. Gupta, C., Singh, Y., Chauhan, D.: Dependency based Process Model for Impact Analysis: A Requirement Engineering Perspective. *International Journal of Computer Applications* 6(6), 28–30 (2010)
6. Sommerville, I.: *Software Engineering*, 6th edn. Addison-Wesley (2001)
7. Ameller, D., Gutiérrez, F., Cabot, J.: Dealing with Non-functional Requirements in Model-Driven Development. In: 18th IEEE International Requirements Engineering Conference (RE) (2010)
8. Nuseibeh, B., Easterbrook, S.M.: Requirements engineering: a roadmap. In: *International Conference on Software Engineering (ICSE)*, pp. 35–46 (2000)
9. Bolchini, D., Mylopoulos, J.: From task-oriented to goal-oriented web requirements analysis. In: *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE 2003)*, p. 166. IEEE Computer Society, Washington, DC, USA (2003)
10. Aguilar, J.A., Garrigós, I., Mazón, J.N., Trujillo, J.: An MDA Approach for Goal-oriented Requirement Analysis in Web Engineering. *J. UCS* 16(17), 2475–2494 (2010)
11. Garrigós, I., Mazón, J.-N., Trujillo, J.: A Requirement Analysis Approach for Using i* in Web Engineering. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) *ICWE 2009*. LNCS, vol. 5648, pp. 151–165. Springer, Heidelberg (2009)
12. Elahi, G., Yu, E.: Modeling and analysis of security trade-offs - a goal oriented approach. *Journal of Data Knowledge Engineering* 68, 579–598 (2009)
13. Aguilar, J.A., Garrigós, I., Mazón, J.-N.: Impact Analysis of Goal-Oriented Requirements in Web Engineering. In: Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., Apduhan, B.O. (eds.) *ICCSA 2011, Part V*. LNCS, vol. 6786, pp. 421–436. Springer, Heidelberg (2011)
14. Aguilar, J.A., Garrigós, I., Mazón, J.N., Trujillo, J.: Web Engineering approaches for requirement analysis- A Systematic Literature Review. In: *6th Web Information Systems and Technologies (WEBIST)*, vol. 2, pp. 187–190. SciTePress Digital Library, Valencia (2010)
15. Schwabe, D., Rossi, G.: The object-oriented hypermedia design model. *Communications of the ACM* 38(8), 45–46 (1995)
16. De Troyer, O.M.F., Leune, C.J.: Wsdm: a user centered design method for web sites. *Comput. Netw. ISDN Syst.* 30(1-7), 85–94 (1998)
17. Casteleyn, S., Woensel, W.V., Houben, G.J.: A semantics-based aspect-oriented approach to adaptation in web engineering. In: *Hypertext*, pp. 189–198 (2007)
18. Fons, J., Valderas, P., Ruiz, M., Rojas, G., Pastor, O.: Oows: A method to develop web applications from web-oriented conceptual models. In: *International Workshop on Web Oriented Software Technology (IWWOST)*, pp. 65–70 (2003)
19. Ceri, S., Fraternali, P., Bongio, A.: Web modeling language (webml): a modeling language for designing web sites. *The International Journal of Computer and Telecommunications Networking* 33(1-6), 137–157 (2000)
20. Escalona, M.J., Aragón, G.: Ndt. a model-driven approach for web requirements. *IEEE Transactions on Software Engineering* 34(3), 377–390 (2008)
21. Koch, N.: The expressive power of uml-based web engineering. In: *International Workshop on Web-oriented Software Technology (IWWOST)*, pp. 40–41 (2002)
22. Bolchini, D., Paolini, P.: Goal-driven requirements analysis for hypermedia-intensive web applications, vol. 9, pp. 85–103. Springer (2004)
23. Molina, F., Pardillo, J., Toval, A.: Modelling Web-Based Systems Requirements Using WRM. In: Hartmann, S., Zhou, X., Kirchberg, M. (eds.) *WISE 2008*. LNCS, vol. 5176, pp. 122–131. Springer, Heidelberg (2008)

24. Yu, E.: Modelling Strategic Relationships for Process Reengineering. PhD thesis, University of Toronto, Canada (1995)
25. Yu, E.: Towards modeling and reasoning support for early-phase requirements engineering. In: RE, pp. 226–235 (1997)
26. Escalona, M.J., Koch, N.: Requirements engineering for web applications - a comparative study. *J. Web Eng.* 2(3), 193–212 (2004)
27. Molina, F., Toval, A.: Integrating usability requirements that can be evaluated in design time into model driven engineering of web information systems. *Adv. Eng. Softw.* 40, 1306–1317 (2009)
28. Horkoff, J., Yu, E.: Evaluating Goal Achievement in Enterprise Modeling – An Interactive Procedure and Experiences. In: Persson, A., Stirna, J. (eds.) PoEM 2009. LNBIP, vol. 39, pp. 145–160. Springer, Heidelberg (2009)
29. Horkoff, J., Yu, E.: A Qualitative, Interactive Evaluation Procedure for Goal-and Agent-Oriented Models. In: CAiSE Forum
30. Garrigós, I.: A-OOH: Extending Web Application Design with Dynamic Personalization. PhD thesis, University of Alicante, Spain (2008)
31. Cachero, C., Gómez, J.: Advanced conceptual modeling of web applications: Embedding operation. In: 21th International Conference on Conceptual Modeling Interfaces in Navigation Design (2002)
32. Aguilar, J.A.: A Goal-oriented Approach for Managing Requirements in the Development of Web Applications. PhD thesis, University of Alicante, Spain (2011)
33. Eclipse (2012), <http://www.eclipse.org/>
34. Pastor, O.: Conference proceedings. In: 1st International Workshop on Web Oriented Software Technology (IWWOST), Valencia, Spain (2001)

Assessing Maintainability Metrics in Software Architectures Using COSMIC and UML

Eudisley Gomes dos Anjos^{1,2}, Ruan Delgado Gomes³, and Mário Zenha-Rela¹

¹ Centre for Informatics and Systems,
University of Coimbra, Coimbra, Portugal
{eudis,mzrela}@dei.uc.pt

² Centre of Informatics,
Federal University of Paraiba, João Pessoa, Brazil

³ Systems and Computer Science Department,
Federal University of Campina Grande, Campina Grande, Brazil
ruan@copin.ufcg.edu.br

Abstract. The software systems have been exposed to constant changes in a short period of time. The evolution of these systems demands a trade-off among several attributes to keep the software quality acceptable. It requires high maintainable systems and makes maintainability one of the most important quality attributes. This paper approaches the system evolution through the analysis of potential new architectures using the evaluation of maintainability level. The goal is to relate maintainability metrics applied in the source-code of OO systems, in particular CCC, to notations defined by COSMIC methods and proposes metrics-based models to assess CCC in software architectures.

Keywords: Maintainability metrics, COSMIC FFP, cohesion, complexity and coupling.

1 Introduction

The software systems are frequently changing because of the technological evolution, the optimization of processes, or because of the integration of existing systems into the development of new software architectures [7]. The higher the complexity of these systems, the higher is the difficulty to maintain them. The effort spent in software maintenance takes a huge amount of the overall software budget. Several surveys state maintenance as one of the most expensive phases in software development, consuming between 60 and 80% of software costs [19].

The constant software changes and the unsuitability to replace existing systems demand that the systems are highly maintainable. A system with this characteristic allows quick and easy changes in the long term. When the system evolves, it is important to keep the maintainability in an acceptable level. Many authors cite the 10 laws of software evolution of Lehman [28] to demonstrate

what happens when the system changes addressing actions that could be taken to avoid them.

Most of the work spent on software evolution focuses on practices towards managing code. However, the evolutionary changes typically comprise structural modifications, which impact the Software Architecture (SA). Modifying and adapting the SA is faster, cheaper and easier than modifying the code. Furthermore, the architectural design can provide a more structured development and quality assurance to accomplish the changes required by the system.

One of the problems concerning analysis of the changes in SA is the lack of metrics. There are many software metrics to assess the maintainability level from the source code [34,5,26]. Nevertheless, when the system requires an evolution and no source code is available, the maintenance analyses becomes more difficult. Moreover, the Common Software Measurement International Consortium (COSMIC) has defined measurements for analysing SA. This paper relates maintainability metrics applied to Object-Oriented (OO) systems, in particular Complexity, Coupling and Cohesion (CCC), to the methods and notations defined by COSMIC and proposes metrics-based models to assess CCC in software architectures.

In section 2 is presented some definitions to insert the reader on the field of this research. The section 3 contains the explanations of object-oriented metrics used to complexity, coupling and cohesion. The variation of these metrics to adapt to architectural designs are presented in the section 4. A case study to show the results of the use of these metrics is presented in section 5 and finally, the section 6, contains the conclusions and future works concerning the continuation of this research.

2 Definitions

This section presents relevant definitions for a better understanding of this paper.

2.1 SA and Evolution

According to IEEE 1471-2000 [20], the software architecture is “*the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution*”.

The relevance of system evolution has been approached in many works [9,31]. There is a need to change software on a constant basis, with major enhancements being made within a short period of time in order to keep up with new business opportunities. So, the approach system evolution is a fundamental element for making strategic decisions, improving system features, and adding economic value to the software.

2.2 Maintainability

The IEEE standard computer dictionary [33] defines maintainability as the ease with which a software system or component can be modified in order to correct faults, improve the quality attributes, or adapt to a changed environment. The modifications may include corrections, improvements or adaptation of the software to changes in environment, requirements and functional specifications.

In ISO/IEC 9126 [25] and Khairudin [22] maintainability is described as a set of sub-characteristics that bear on the effort needed to make specified modifications such as: modularity, flexibility, adaptability, portability and so forth. These sub-characteristics are refined to attributes used to evaluate the level of maintainability in a system. Among these attributes we can mention complexity, composability, cohesion, coupling, decomposability, etc.

A system with high level of maintainability has low cost for its maintenance activities. It reduces the total maintenance cost or improves the system by performing more maintenance activities at the same total cost. The level of maintainability can be measured applying metrics over its attributes.

2.3 COSMIC

COSMIC is a world-wide group of metric experts which has developed methods of measuring a functional size of software. The COSMIC Full Function Point uses Functional User Requirements (FUR) to measure the software size. It also quantifies the software's sub-processes (data movements) within each process, measuring the functionality of the software [15].

The use of COSMIC in this work is due to unify the architectural notation to be used by the metrics. It is a relevant characteristic for the project this work is inserted in, which will be continued and improved. With the use of COSMIC data movements it is possible to classify the processes using a standard notation. The main data movements defined in COSMIC are:

E: Entry

X: Exit

R: Read

W: Write

N: Number of components

L: Layer

3 Object-Oriented Metrics

A software metric is used to measure some property of a piece of software. They aim to evaluate the ability of software to achieve predefined objectives.

There are many metrics used to evaluate the maintainability attributes in OO systems [14,8]. This paper focuses on three essential attributes: complexity, coupling and cohesion (CCC). Their metrics can be applied during various software development phases and are used to evaluate the quality of software [17]. Thus, the use of these metrics can provide complementary solutions that are potentially useful for architecture evaluation [8], leading to more secured and dependable systems [13]. The next subsections describe the CCC attributes and the metrics approached in this paper.

3.1 Complexity

In IEEE definition, complexity is the degree to which a system or component has a design or implementation that is difficult to understand and verify. The definition argues that the complexity is a property of the design implementation, i.e. source code or design. On the other hand, this definition of complexity also concerns the effort needed to understand and verify the design implementation. This means that two different entities are involved in the definition, process (effort) and product (design or source code) [21].

This work concerns to complexity related to the product. So, it assess the evolution of structural design and the impact of software changes in the code. In software systems, the more complex are the structures (or modules), the harder is to understand, change, reuse, produce and easier to have a defect. The complexity metrics approached here are CCN and FFC.

Cyclomatic Complexity Number (CCN): It is one of the oldest complexity metrics developed by McCabe [29]. The essence of this metric is to count the number of linearly independent and executable paths through a module. The formula to find the CCN is defined as:

$$CCN = (E - N) + 2P,$$

Where:

E : Number of edges

N : number of nodes (vertices)

P : number of independent graphs (usually $P = 1$)

Fan-in Fan-out Complexity (FFC): Henrys and Kafura [23] identify a metric using fan-in and fan-out measures, also called information flow. This metric measures the level of complexity in a system module. The method suggest identifying the number of calls from a module and the number of calls to the module. In the complexity measurement it is also used the length of the module which may be substituted by the CCN number. The fan-in and fan-out metrics by themselves can be considered by some authors as coupling metrics stating values for

afferent and efferent coupling respectively. The complexity is determined by the follow formula:

$$FFC = L * (I * O)^2,$$

Where:

L : length of the module (number of lines of code or CCN)

I : Number of calls to the module (fun-in)

O : Number of calls from the module (fun-out)

3.2 Coupling

Coupling is one of the attributes of modularity with most influence on software maintenance as it has a direct effect on maintainability. Coupling metrics are used in tasks such as impact analysis [8], assessing the fault-proneness of classes [35], fault prediction [16], re-modularization [2], identifying of software components [27], design patterns [3], assessing software quality [8], etc. In general, the main goal in the software design is get the lower coupling as possible.

In OO, classes that are strongly coupled are affected by changes and bugs in other classes [30]. Modules with high coupling have a substantial importance for software architectures and thus need to be identified.

This work approaches two coupling metrics: CBO and RFC.

Coupling Between Object (CBO): It is one of the metrics proposed by Chidamber and Kemerer (CK) [24] used to measure the coupling between classes. The initial method, according to the specification of the authors, uses two measures: the number of classes within a package and the relationships between these classes with others in different packages. Later, other factors were considered such as: number of methods in the classes relationships, inheritance relationships between classes, etc. Generically the level of CBO in a package can be determined by the formula showed below.

$$CBO = \frac{NL}{NC}$$

Where,

NL = number of links

NC = number of classes

Response For Class (RFC): RFC is one more metric of CK and calculates the number of distinct methods and constructors invoked by a class. It is the number of methods in a particular class plus the number of methods invoked in other classes. Each method is counted just once, even if that method has many relationships with methods in other classes. The formula to calculate RFC is described below.

$$RFC = M + NC$$

Where,

M = number of method in the analysed class

NC = number external called methods

3.3 Cohesion

A module has high cohesion when the relationships among its elements are tight and the module provides a single functionality. The higher is the module cohesion, the easier the module is to develop, maintain, and reuse. Also, there is empirical evidence that supports the importance of cohesion in structured design [10]. Therefore, virtually every software engineering text describes high cohesion as a very desirable property of a module [12].

Numerous cohesion metrics have been proposed for object oriented modules [8, 11], and researchers have attempted to take the characteristics of methods into account in the definition of cohesion metrics [12]. Nevertheless, there is a lack of these methods in software architecture designs hampering the application of metrics. One reason is due to the architecture not contains the internal content of a component. The cohesion metrics approached here are: TCC and LCC.

Tight and Loose Class Cohesion (TCC and LCC): The metrics of TCC and LCC provide a way to measure the cohesion of a class. For TCC and LCC only visible methods are considered (methods that are not private). TCC represents a density of attribute-sharing relationships between public methods in a class. LCC represents extended relationships which are constructed by the transitive closure of attribute-sharing relationships. The higher is TCC and LCC, the more cohesive the class is [6].

$$TCC(C) = \frac{NDC(C)}{NP(C)}$$

$$LCC(C) = \frac{NDC(C)+NIC(C)}{NP(C)}$$

Where,

$$NP(C) = \frac{N*(N-1)}{2}$$

N: number of methods

NDC(C): Number of direct connectivity between methods.

NIC(C): Number of indirect connections between methods.

4 Architectural Metrics

The metrics approached in the previous section are used to evaluate source-code of object-oriented systems. This work aims to apply these metrics over SA designs. Usually, the architectural designs are described using Architecture Description Languages (ADLs). These languages describe a system at the higher levels of abstraction and are intended to be both human and machine readable, although an architect interpretation can differ of other architect. Depending on the ADL used, software architectures can be designed in different ways.

The software architecture is the best element to manipulate when a system evolution is required. The analysis of the changes is simpler, easier and cheaper to be done using architectural modifications. Moreover, the evaluation of the quality attributes may be performed. It allows verify if the system has an acceptable level of maintainability.

There are some works which approach the impact of software changes in software architectures. According to [4] it is possible to define metrics to abstract levels of the system like software architectures. More than that, he discusses the possibilities to use OO metrics for software architectures and support this paper to understand the viability in applying such metrics.

The work presented in [32] demonstrates how the evolution can decrease the level of maintainability of the system. It establishes the need of quality evaluation before coding the changes for the system evolution. In [18] a tool called *Ævol* was developed to define and plan the evolution of systems using the architectural model.

In [36] Zayaraz approaches the use of COSMIC to define architectural metrics to evaluate quality attributes. His work uses sub-characteristics of maintainability as the target of metrics. However, this work does not explain exactly how the metrics can be applied. Besides, is missing to propose a validation model to define the quantification and limits of these metrics.

The work presented here relates maintainability metrics applied over source-code of OO systems, in particular the CCC metrics mentioned before, to notations defined by COSMIC methods. So, it adapts the COSMIC notations to UML diagrams and proposes metrics-based models to evaluate software architectures. Each UML diagram can be considered as an architectural layer and the COSMIC data movements are linked to diagram elements.

The case study carried out in this paper considers the component diagram as the main system design description since it is one of the closest to other ADLs diagrams. Considering the elements of a component diagram and the COSMIC data movements, the elements adopted are described below.

E: Dependencies from the component.

X: Dependencies to the component.

W: Interfaces provided by the component.

R: Interfaces used by the component.

N: Number of components in a diagram.

L: Layer (in this case UML diagram)

S: Number of sub-components in a component

Using the considerations mentioned above the metrics are adapted to UML as following:

Complexity

$$CCN = ((E + X + W + R) - N) + 2L$$

$$FFC = CCN * ((E + W) * (X + R))^2$$

Coupling

$$CBO = \frac{NL}{N}$$

$$RFC = S + E + R$$

Where:

NL = Number of links of the component ($E + X + W + R$)

Cohesion

$$TCC(C) = \frac{E+X+W+R}{P(S)}$$

$$LCC(C) = \frac{(E+X+W+R)+NIC(S)}{NP(C)}$$

Where:

$$P(S) = \frac{S*(S-1)}{2}$$

$NIC(S)$ = Number of indirect connections between components

It is important to mention that each metric has a scope of application. For example, in the complexity metric CCN a component which posses sub-components is considered as only one element for the layer under evaluation. It has the whole diagram as scope of application without considering the internal relations of the component. Nevertheless, in the cohesion metrics the elements of evaluation is the component itself, so, in this case, the sub-components are relevant to the measurement. If the component has no subcomponents ($S = 1$) the calculation of TCC and LCC is not applied because it has the maximum cohesion. The Table 1 shows the metrics approached and the scope they are applied.

Table 1. Metrics and scope of application

Metric	Scope
CCN	Whole diagram
FFC	Component
CBO	Component
RFC	Component
TCC	Component
LCC	Component

5 Case Study

The initial case study uses an open-source software, the DCMMS, to validate the hypotheses of architectural metrics application for maintainability. The DC Maintenance Management System (DCMMS) is a web-based system used to store, manage and analyse customer complaints for water and waste-water networks. A work order with map is created for every complaint. After completion of the works in the field, updated information from the work order is entered to the system for further analysis. This system was chosen because it is small, open-source, well documented, and therefore more didactic to the study approached in this paper. The Figure 1 shows the UML component diagram for DCMMS version 1.1.2.

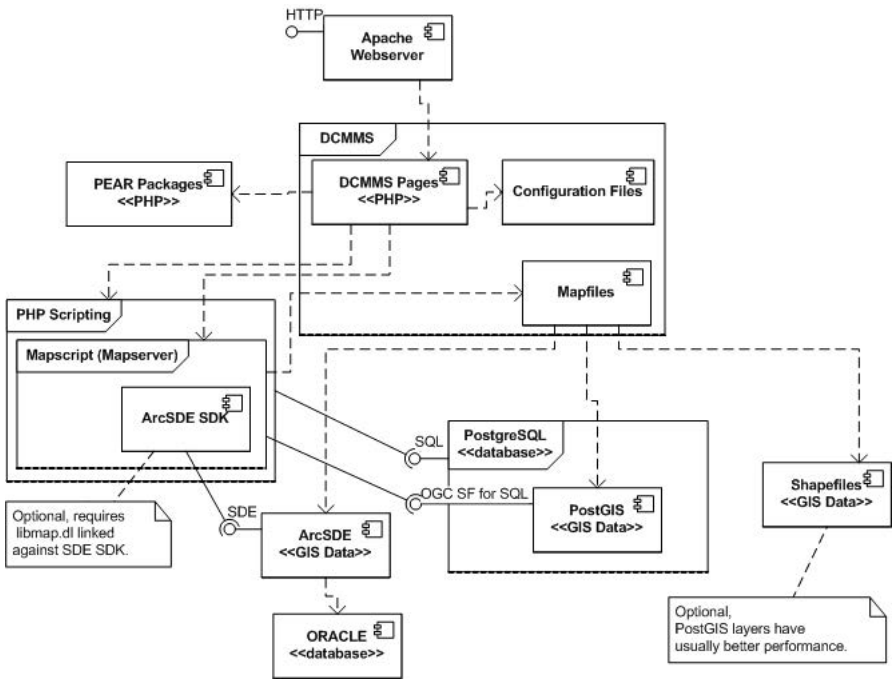


Fig. 1. DCMMS component diagram (source: <http://dcmmms.sourceforge.net/>)

This diagram is transformed in a graph or converted to a file with XMI (XML Metadata Interchange) extension. It reduces the complexity of the diagram for large scale systems and allows automatize the measurement process. The graph obtained for the DCMMS component diagram is presented in Figure 2. This graph allows a faster view over the diagram limiting to the perception of the useful elements and connections to the layer over evaluation.

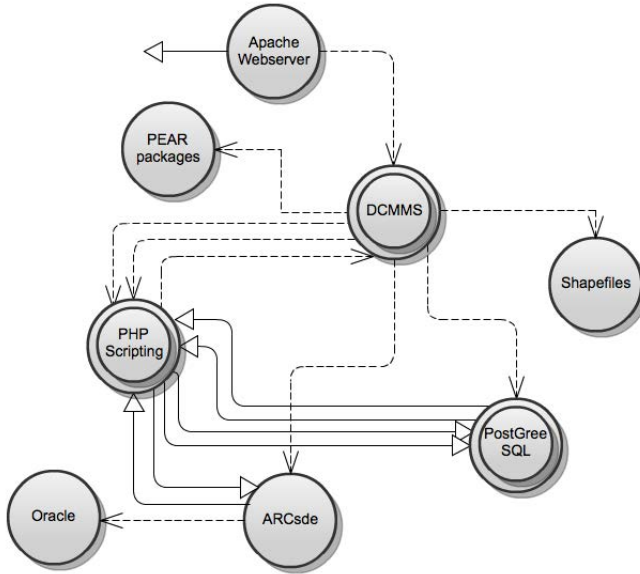


Fig. 2. DCMMS graph transformed to evaluation

The CCC metrics were applied over DCMMS in the whole component diagram and for each different component, depending on the scope. The results are showed in the Table 2. Each column refers to a CCC metric and the lines to the components. The last line, AVG, includes the average value among all components in the diagram for a specific metric.

Table 2. Comparison between OO elements and Architectural elements

	FFC	CBO	RFC	TCC	LCC
Apache Webserver	10	0.25	3	not applied	not applied
PEAR packages	0	0.125	1	not applied	not applied
DCMMS	1440	1	9	2.7	8.3
Shapefiles	0	0.125	1	not applied	not applied
PHP Scripting	4000	1.125	5	not applied	not applied
Oracle	0	0.125	1	not applied	not applied
ARCSde	40	0.375	3	not applied	not applied
Postgree SQL	360	0.625	3	not applied	not applied
AVG	731.25	0.47	3.25	not applied	not applied

In way to improve the understand of results, some considerations should be done. In the measurement of components with only entrances connections and no exit or only exits with no entrance, the coupling is very small and it is not captured by FCC metric. In these cases, $FFC = 0$.

There are many components where the application of TCC and LCC is not adaptable. It is due to the lack of sub-components inside the component and hence, the cohesion is maximum.

Among the many tests we can do over the diagram one of them can demonstrate how to improve the level of maintainability. The test consist in substitute the component named DCMMS by its subcomponents, increasing the size of the component diagram. The Figure 3 shows this alternative system. Although noticeably the diagram is more complex, the measurement of CCC in this DCMMS evolution (or variation) can show better results.

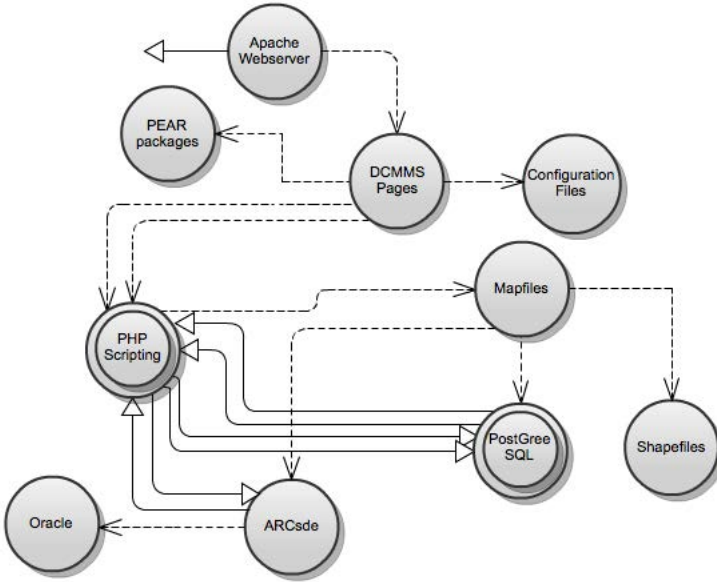


Fig. 3. DCMMS graph after remove DCMMS sub-components

It is possible to see that, although the aspect of the system looks more complex, the level of complexity, cohesion and coupling improved with this modification letting to a more maintainable system as it is showed in Table 3. Nonetheless, the changes do not always let all the attributes to be better or worst. Sometimes is important realize the trade-off between the attributes to reach the most appropriate architecture.

The metrics used here validate the hypothesis in apply them in architectural designs but, the final values presented here are analysed using the OO metrics limits. It is still necessary to define the limits of the metrics considering architectural issues to understand better the level of quality.

Table 3. Comparison for the modified architecture

	FFC	CBO	RFC	TCC	LCC
Apache Webserver	9	0.2	3	not applied	not applied
PEAR packages	0	0.1	1	not applied	not applied
DCMMS pages	144	0.5	5	not applied	not applied
Configuration Files	0	0.1	1	not applied	not applied
Mapfiles	81	0.4	4	not applied	not applied
Shapefiles	0	0.1	1	not applied	not applied
PHP Scripting	3600	0.9	5	not applied	not applied
Oracle	0	0.1	1	not applied	not applied
ARCsdde	36	0.3	3	not applied	not applied
Postgree SQL	324	0.5	3	not applied	not applied
AVG	419.4	0.32	2.7	not applied	not applied

6 Conclusions and Future Works

The difficulty and cost to modify the code and evaluate the quality for a possible system evolution demonstrates the need for such architectural evaluation. As the metrics for codes, architectural metrics must be general enough to works in different kinds of systems. Also, more than one evolution can be compared, helping the architect in tuning the required design. In way to get this goal, this work might still be refined and improved.

This paper is part of a Ph.D. thesis about assessing maintainability in software architectures. The work presented here is an initial approach to demonstrate the viability to assess maintainability in software architectures using well-known metrics. The results show that such metrics can be applied over SA, although some changes might be required. Moreover, relevant issues were raised with this study supporting the authors in the definition of future works to continue this research.

Some of the guidelines for further work are:

Definition of Metric Limits. It is important to define the limits for the metrics applied over software architectures. It may be done measuring different systems versions with several levels of CCC to understand the behaviour of the changes and compare to source-code results. It allows to know how maintainable the system is based on the values obtained by the metrics.

Inter-diagrams Metrics. The results for DCMMS system, show that complexity and coupling seems to be more measurable than cohesion. It happens due to the lack of information in the layer we take as example. If the metrics are applied over other diagram, different results can be obtained. Thus, using more than one layer, inter-diagram information can be used to improve the results.

Definition of New Metrics. The evaluation of different system versions and different UML diagrams provides a new amount of data. It enables the assessment of existing metrics and the definition of new ones. Also, the comparison between the level of maintainability in code with the modifications performed in the system helps to realize the impacts of changes over the levels of complexity, cohesion and coupling.

Automate the Maintainability Evaluation. When the architectural metrics have been accomplished, the evaluation process to support the software architect to tune the architecture design using a simple interface should be automated. It allows to compare the maintainability level for different evolutions of the system and chose the best design according to the requirements. This study was initiated in [1] and drives the development of such architectural tool.

The future works proposed here has as target large scale systems. The process for development of this work contains three main phases. The first phase consists in applying the metrics over source-code of several versions of a group of systems to analyse the behaviour of the results during its evolution. In the second phase the architectures of these systems are analysed using the same metrics over UML diagrams, as initially proposed here. Lastly, in the third phase, the final metrics are defined, adapted to architectural designs and implemented in a set of tools to assist the software architect.

References

1. Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., Apduhan, B.O. (eds.): ICCSA 2011, Part V. LNCS, vol. 6786. Springer, Heidelberg (2011)
2. e Abreu, F.B., Pereira, G., Sousa, P.: A coupling-guided cluster analysis approach to reengineer the modularity of object-oriented systems. In: Proceedings of the Conference on Software Maintenance and Reengineering, CSMR 2000, p. 13. IEEE Computer Society, Washington, DC, USA (2000)
3. Antoniol, G., Fiutem, R., Cristoforetti, L.: Using metrics to identify design patterns in object-oriented software. In: Proceedings of the 5th International Symposium on Software Metrics, METRICS 1998, p. 23. IEEE Computer Society, Washington, DC (1998)
4. Bengtsson, P.: Towards maintainability metrics on software architecture: An adaptation of object-oriented metrics. In: First Nordic Workshop on Software Architecture, NOSA 1998 (1998)
5. Berns, G.M.: Assessing software maintainability. *Commun. ACM* 27, 14–23 (1984)
6. Bieman, J.M., Kang, B.-K.: Cohesion and reuse in an object-oriented system. *SIGSOFT Softw. Eng. Notes* 20, 259–262 (1995)
7. Bode, S.: On the role of evolvability for architectural design. In: Fischer, S., Maehle, E., Reischuk, R. (eds.) *GI Jahrestagung. LNI*, vol. 154, pp. 3256–3263. GI (2009)
8. Briand, L.C., Bunse, C., Daly, J.W.: A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs. *IEEE Trans. Softw. Eng.* 27, 513–530 (2001)

9. Cai, Y., Huynh, S.: An evolution model for software modularity assessment. In: Proceedings of the 5th International Workshop on Software Quality, WoSQ 2007, p. 3. IEEE Computer Society, Washington, DC, USA (2007)
10. Card, D.N., Church, V.E., Agresti, W.W.: An empirical study of software design practices. *IEEE Trans. Softw. Eng.* 12, 264–271 (1986)
11. Chae, H.S., Kwon, Y.R., Bae, D.-H.: A cohesion measure for object-oriented classes. *Softw. Pract. Exper.* 30, 1405–1431 (2000)
12. Chae, H.S., Kwon, Y.R., Bae, D.H.: Improving cohesion metrics for classes by considering dependent instance variables. *IEEE Trans. Softw. Eng.* 30, 826–832 (2004)
13. Chowdhury, I., Zulkernine, M.: Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities. *J. Syst. Archit.* 57 (March 2011)
14. Daly, J., Brooks, A., Miller, J., Roper, M., Wood, M.: Evaluating inheritance depth on the maintainability of object-oriented software. *Empirical Software Engineering* 1(2), 109–132 (1996)
15. Efe, P., Demirors, O., Gencel, C.: Mapping concepts of functional size measurement methods. In: *COSMIC Function Points Theory and Advanced Practices*, pp. 1–16 (2006)
16. El Emam, K., Melo, W., Machado, J.C.: The prediction of faulty classes using object-oriented design metrics. *J. Syst. Softw.* 56, 63–75 (2001)
17. Fenton, N.E.: *Software Metrics: A Rigorous and Practical Approach*. International Thomson Computer Press, Boston (1996)
18. Garlan, D., Schmerl, B.R.: A tool for defining and planning architecture evolution. In: *ICSE*, pp. 591–594. IEEE (2009)
19. Glass, R.L.: *Facts and Fallacies of Software Engineering*. Addison-Wesley (2002)
20. IEEE Architecture Working Group. IEEE std 1471-2000, recommended practice for architectural description of software-intensive systems. Technical report. IEEE (2000)
21. Habra, N., Abran, A., Lopez, M., Sellami, A.: A framework for the design and verification of software measurement methods. *J. Syst. Softw.* 81, 633–648 (2008)
22. Hashim, K., Key, E.: *Malaysian Journal of Computer Science* 9(2) (1996)
23. Henry, S., Kafura, D.: Software structure metrics based on information flow. *IEEE Transactions on Software Engineering* 7(5), 510–518 (1981)
24. Hitz, M., Montazeri, B.: Chidamber and kemerer’s metrics suite: A measurement theory perspective. *IEEE Trans. Softw. Eng.* 22, 267–271 (1996)
25. ISO. International Standard - ISO/IEC 14764 IEEE Std 14764-2006. ISO/IEC 14764:2006 (E) IEEE Std 14764-2006 Revision of IEEE Std 1219-1998), pp. 1–46 (2006)
26. Kuipers, T.: Software Improvement Group, and Joost Visser. Maintainability index revisited - position paper. *Complexity*, 5–7 (2007)
27. Lee, J.K., Seung, S.J., Kim, S.D., Hyun, W., Han, D.H.: Component identification method with coupling and cohesion. In: *Proceedings of the Eighth Asia-Pacific on Software Engineering Conference, APSEC 2001*, p. 79. IEEE Computer Society, Washington, DC, USA (2001)
28. Lehman, M.M.: Laws of Software Evolution Revisited. In: Montangero, C. (ed.) *EWSP* 1996. LNCS, vol. 1149, pp. 108–124. Springer, Heidelberg (1996)
29. McCabe, T.J.: A complexity measure. *IEEE Transactions on Software Engineering* 2, 308–320 (1976)
30. Poshyvanyk, D., Marcus, A.: The conceptual coupling metrics for object-oriented systems. In: *Proceedings of the 22nd IEEE International Conference on Software Maintenance*, pp. 469–478. IEEE Computer Society, Washington, DC (2006)

31. Rowe, D., Leaney, J.: Evaluating evolvability of computer based systems architectures - an ontological approach. In: Proceedings of the 1997 International Conference on Engineering of Computer-Based Systems, ECBS 1997, pp. 360–367. IEEE Computer Society, Washington, DC (1997)
32. Shen, H., Zhang, S., Zhao, J.: An empirical study of maintainability in aspect-oriented system evolution using coupling metrics. In: Proceedings of the 2008 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering, pp. 233–236. IEEE Computer Society, Washington, DC (2008)
33. The Institute of Electrical and Eletronics Engineers. IEEE standard glossary of software engineering terminology. IEEE Standard (1990)
34. Thongmak, M., Muenchaisri, P.: Maintainability metrics for aspect-oriented software. *International Journal of Software Engineering and Knowledge Engineering IJSEKE* 19(3), 389–420 (2009)
35. Yu, P., Systä, T., Müller, H.A.: Predicting fault-proneness using oo metrics: An industrial case study. In: Proceedings of the 6th European Conference on Software Maintenance and Reengineering, CSMR 2002, pp. 99–107. IEEE Computer Society, Washington, DC (2002)
36. Zayaraz, G., Thambidurai, P., Srinivasan, M., Rodrigues, P.: Software quality assurance through cosmic ffp. *ACM SIGSOFT Software Engineering Notes* 30(5), 1 (2005)

Plagiarism Detection in Software Using Efficient String Matching

Kusum Lata Pandey¹, Suneeta Agarwal², Sanjay Misra³, and Rajesh Prasad⁴

¹Ewing Christian College, Allahabad, India

²Motilal Nehru National Institute of Technology, Allahabad, India

³Atilim University, Ankara, Turkey

⁴Ajay Kumar Garg Engineering College, Ghaziabad, India

klpandey@gmail.com, suneeta@mnnit.ac.in, smisra@atilim.edu.tr,
rajeshpd18@yahoo.com

Abstract. String matching refers to the problem of finding occurrence(s) of a pattern string within another string or body of a text. It plays a vital role in *plagiarism detection* in software codes, where it is required to identify similar program in a large populations. String matching has been used as a tool in a software metrics, which is used to measure the quality of software development process. In the recent years, many algorithms exist for solving the string matching problem. Among them, Berry–Ravindran algorithm was found to be fairly efficient. Further refinement of this algorithm is made in TVSBS and SSABS algorithms. However, these algorithms do not give the best possible shift in the search phase. In this paper, we propose an algorithm which gives the best possible shift in the search phase and is faster than the previously known algorithms. This algorithm behaves like Berry-Ravindran in the worst case. Further extension of this algorithm has been made for parameterized string matching which is able to detect plagiarism in a software code.

Keywords: String matching, plagiarism detection, bad character shift, parameterized matching and RGF.

1 Introduction

String matching refers to the problem of finding occurrence(s) of a pattern string within another string or body of text. String matching is a very important subject in the wider domain of text processing. The main application of string matching includes: text editor(s), information retrieval, image processing, computational biology, pattern recognition etc. Also, it plays a vital role in *plagiarism detection* in software codes, where it is required to identify similar program in a large populations. String matching has been used as a tool [1, 2] in a software metrics, which is used to measure the quality of a software development process. Another type of string matching called parameterized string matching (p-match) has been used in [4, 7] for finding duplicated code in a large software text. This type of string matching is able to find similar codes where systematic renaming of identifiers and variable is done.

Fig.1 given below illustrates the three major processes that make up generalized plagiarism detection system [2]. String matching (or p-match) can be used in software analyzer to produce the list of program pair with a measure of differences between their metric representations. The final phase orders the list on metric similarity and filters out those pairs exhibiting significant metric differences. The submissions listed as potential plagiarisms are ultimately retrieved for manual examination and classification.

Since string matching plays an important role in plagiarism detection, therefore we mainly focus on developing efficient algorithm for solving the string matching problem. In the recent years [3, 5, 6, 8, 9, 11], many efficient algorithms exist for solving string matching problem. Among them, Berry-Ravindran [6] algorithm is found to be fairly efficient. Further refinement of this algorithm is made in TVSBS [13] and SSABS [8] algorithms. However, these algorithms do not give the best possible shift in the search phase. In this paper, we propose an algorithm which gives the best possible shift in the search phase and is faster than the previously known algorithms. This algorithm behaves like Berry-Ravindran in the worst case. Further extension of this algorithm has been made for parameterized string matching which is able to detect plagiarism in a software code.

The paper is organized as follows. Sec. 2 presents related algorithms. Sec. 3 presents the proposed algorithm. Sec. 4 presents the experimental results. Finally, we conclude in Sec. 5.

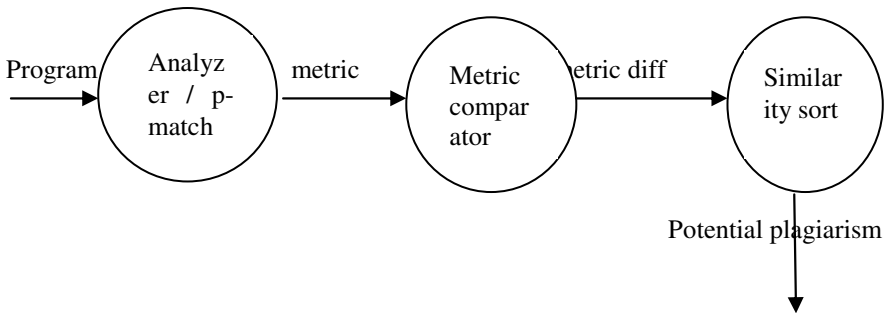


Fig. 1. A Plagiarism Detection System

2 Related Algorithms

In this section, we present some related concepts and algorithms. Thorough out the paper, we assume that the pattern $P[0..m-1]$ and text $T[0..n-1]$ are array of m and n characters where $m \leq n$. Symbols of pattern and text are drawn from some alphabet Σ of size σ .

2.1 Sunday's Quick Search Algorithm

Sunday [5, 12] simplifies the Boyer-Moore [3] algorithm further to derive new algorithm: quick search. It uses only bad character shift to shift the window. At any moment, if the window is positioned on the text factor $T[j\dots j+m-1]$, $0 \leq j \leq n-m$, the bad character shift is determined by text character $c = T[j+m]$ as follows:

$$qsBc[c] = \begin{cases} \min\{i \mid 1 \leq i \leq m, P[m-i] = c\} & \text{if } c \in P \\ m+1 & \text{otherwise} \end{cases}$$

In the preprocessing phase, running time of the algorithm is $O(m+\sigma)$ and space complexity is $O(\sigma)$. In the searching phase, running time of the algorithm in average and worst case is: $O(mn)$ and in best case is: $O(n/m)$.

2.2 Raita Algorithm

Raita [10, 12] developed an algorithm which at each alignment of the pattern $P[0\dots m-1]$ on the text window $T[j\dots j+m-1]$, $0 \leq j \leq n-m$, first compares the last character of the pattern with the rightmost text character of the window, if they match, then it compares the first character of the pattern with the leftmost character of the window. In case of successful match, it compares the middle character of the pattern with middle text character of the window. And finally if they match, it actually compares the remaining characters from the second to the last. Shifts in this algorithm are decided by rightmost character of m -length text window of the current alignment.

Running time of the algorithm in preprocessing phase is $O(m+\sigma)$ and space complexity is $O(\sigma)$. In the searching phase, running time of the algorithm in average and worst case is: $O(mn)$ and in best case is: $O(n/m)$.

2.3 Berry-Ravindran Algorithm

Berry and Ravindran [6, 12] designed an algorithm which performs the shifts by considering the bad-character shift for the two consecutive text characters immediately to the right of the m -length text window.

Preprocessing phase of the algorithm consists in computing for each pair of characters (a, b) with $a, b \in \Sigma$, the rightmost occurrence of ab in axb .

$$brBc[a, b] = \min \begin{cases} 1 & \text{if } x[m-1] = a \\ m-i+1 & \text{if } x[i] x[i+1] = ab \\ m+1 & \text{if } x[0] = b \\ m+2 & \text{otherwise} \end{cases}$$

The preprocessing phase is in $O(m+\sigma)$ space and time complexity.

After an attempt where the window is positioned on the text factor $T[j \dots j+m-1]$, a shift of length $\text{brBc}[T[j+m], T[j+m+1]]$ is performed. The text character $T[n]$ is equal to the null character and $T[n+1]$ is set to this null character in order to be able to compute the last shifts of the algorithm.

The searching phase of the Berry-Ravindran algorithm has a $O(mn)$ time complexity.

2.4 TVSBS Algorithm

The TVSBS [13] algorithm is basically a blend of Berry-Ravindran, Raita and Quick Search algorithm. The order of comparisons is carried out by comparing the last character of the m -length text window and that of the pattern first and once they match, the algorithm further compares the first character of this text window and that of the pattern. This establishes an initial resemblance between the pattern and the window. The remaining characters are then compared from right to left until a complete match or a mismatch occurs. After each attempt, shift of the window is gained by brBc shift value for the two consecutive characters immediately next to the window. The brBc function has been exploited to obtain the maximal shift and this reduces the number of character comparisons.

2.5 Parameterized String Matching

In 1992, Baker [7] first time introduces parameterized matching as a model for identifying duplicated code in a software code. A given pattern $P[0 \dots m-1]$ is said to parameterized match (p -match) with a substring t of the text $T[0 \dots n-1]$, if there exists a bijection from the symbols of P to the symbols of t . In this problem, two disjoint alphabets Σ of size σ and Π of size π (either one may be empty) are used. Σ is used for fixed symbols and Π is used for parameterized symbols. In the bijection map used for p -match, the symbols of Σ must match exactly whereas the symbols of Π may be renamed. Searching of bijection has been made easy by Baker [7] with the proposed concept of *prev*-encoding (*prev*). *Prev*(S) of a string S assigns a non-negative integer p to each occurrence of a parameter symbol s ($s \in \Pi$) in S , where p is the number of symbols since the previous occurrence of s in S . *Prev* of first occurrence of any parameter symbol is taken as 0. For example, $\text{prev}(XYABX) = 00AB4$ and $\text{prev}(ZWABZ) = 00AB4$. Here, strings $XYABX$ and $ZWABZ$ are p -matched with the bijective mapping $X \leftrightarrow Z$ and $Y \leftrightarrow W$. It is to be noted that their *prev*-encodes match exactly. Thus one way of finding p -match is through *prev*-encoding. However, for large text, direct *prev*-encoding may not work [7]. Baker in [7] has proposed a method which can overcome the problem.

In [14], Boyer-Moore algorithm has been extended to parameterized string matching for finding duplicate code in a large software text. In [15], a fast parameterized Boyer-Moore algorithm using q -gram has been developed. This algorithm uses restricted growth function (RGF) for encoding the text / pattern. A single alphabet called parameterized alphabet has been used in this algorithm. This algorithm was shown to be better than Boyer-Moore algorithm of [14].

2.6 Parameterized Sunday (PSUNDAY) Algorithm

This section presents the parameterized version of Sunday's quick search algorithm [16]. In the pre-processing phase of this algorithm, we first form $m-q+1$ overlapping q -grams of the pattern $P[0\dots m-1]$, where $q (>0)$ is the size of the q -gram. We then take prev-encoding of each q -gram. An integer equivalent (for indexing) of each q -gram is obtained as in section 3.4.2. These integer equivalent values are concatenated in order to form a new pattern. The bad character shift is computed from this new pattern.

Let the text be represented as: $T_0, T_1, T_2, \dots, T_i, T_{i+1}, \dots, T_{n-1}$ and last character of the m -length window at any instant is T_i ($m-1 \leq i \leq n-1$). In the searching phase, we first calculate integer equivalent (as earlier) of the last q -gram of this m -length text window. If it matches with last q -gram (integer equivalent) of the pattern, then same process of matching is repeated one-by-one with preceding q -grams of the text window. In case, all q -grams of the text window are matched, then the corresponding valid shift is reported and window is shifted by "SHIFT". In the case of mismatch at any point, window is shifted by the same shift "SHIFT". This shift "SHIFT" is calculated by using the q -gram ending with character T_{i+1} . This procedure is followed until end of the text. Example given below illustrates the algorithm.

Example

Let the pattern $P = \text{"abaaba"}$ and text $T = \text{"bbabbababa"}$ and $q = 3$. We assume that all the symbols of P and T are parameterized symbols. The 3-grams of the pattern are: aba, baa, aab, and aba. Prev-encodings of pattern are: 002, 001, 010, and 002. By reserving enough bits for each character, the integer equivalent (index) of each q -gram are: $(0010 \equiv 2, 0001 \equiv 1, 0100 \equiv 4, \text{ and } 0010 \equiv 2)$ respectively. The new pattern becomes "2142". Table 1 shows the Sunday's pre-processing table.

Table 1. Sunday's (Quick search) Bad character (qsBC) Table (* is meant to represent any other character)

Char (c)	'2'	'1'	'4'	*
qsBC [c]	1	3	2	5

Steps of the searching part:

$T = [b \ b \ a \ b \ b \ a] \ b \ a \ b \ a$.

Using the 3-gram "bab", shift is first calculated: the integer equivalent of this 3-gram (bab) is 2, hence the SHIFT is 1 ($=\text{qsBC}[2]$).

Start with the last 3-gram (i.e. bba) of the window. Integer equivalent of prev-encoded of this 3-gram is 4. This character (i.e. 4) is compared with last character (i.e. 2) of the new form of the pattern "2142". There is a mismatch (i.e. $4 \neq 2$). Hence there is no need to consider the preceding 3-grams.

$T = b \ [\ b \ a \ b \ b \ a \ b] \ a \ b \ a$

Integer equivalent of last 3-gram (i.e. bab) is 2, which is equal to last character of the pattern “2142”. Hence we repeat the process for preceding 3-grams and so on. Since there is a match for each 3-gram, therefore pattern (abaaba) parametrically occurs (with bijective mapping $a \leftrightarrow b$) in the text with valid shift 1.

3 The Proposed Algorithm

The algorithm proposed here aims to maximize the shift value for each iteration as well as decrease the time during the searching phase. The proposed algorithm is cross between Berry-Ravindran and TVSBS algorithm. The Berry-Ravindran bad character function (brBc) is found to be the most effective in calculating the shift value but many times even brBc doesn't give the maximum shift possible. In this paper, we propose an improvement in the shift. The searching phase is same as in TVSBS algorithm.

3.1 Pre-processing Phase

This phase is performed by using *two bad* character functions. This provides maximum shift most of the times and its worst case is the shift provided by brBc. Pre-processing phase of the algorithm consists of computing for each pair of characters (a, b) for all $a, b \in \Sigma$, the rightmost occurrence of *ab* in the pattern. First we consider the pair of characters lies immediately after the window and calculate the bad character according to them, as done in Berry-Ravindran algorithm known as brBc. It is defined as:

$$\text{brBc}[a, b] = \min \begin{cases} 1 & \text{if } x[m-1] = a \\ m-i+1 & \text{if } x[i]x[i+1] = ab \\ m+1 & \text{if } x[0] = b \\ m+2 & \text{otherwise} \end{cases}$$

Now it is possible that the shift provided by brBc may sometimes not be the best shift. Here we propose a new function named as New Bad Character function (newBc) which sometimes gives better shift than brBc. newBc is computed by taking the pair of the last character of the *m*-length text window and the one next to it outside the window. It is defined as:

$$\text{newBc}[a, b] = \min \begin{cases} m-i & \text{if } x[i]x[i+1]=ab, \\ m & \text{if } x[0]=b; \\ m+1 & \text{otherwise.} \end{cases}$$

Now the maximum shift of both the shift algorithms is taken as valid shift for the proposed algorithm, i.e. $\text{shift} = \max(\text{brBc}[a, b], \text{newBc}[a,b])$. For efficiency purposes the shift for all pair of alphabets for both brBc and newBc is stored in 1D-arrays rather than 2D-arrays, which decreases the access time. In this algorithm, we have used brBc over Quick Search bad character and Boyer–Moore bad character because

it gives the better shift most of the times as the probability of the rightmost occurrence of ab in the pattern is less than the probability of a .

3.2 Searching Phase

Searching is done as in the TVSBS [13] algorithm. It includes three stages.

Step 1: In stage 1, the last character of the m -length text window is compared to the last character in pattern. If it matches than we compare the first character of window with first character of the pattern. If it matches too, than we move to the stage 2 otherwise we jump to stage 3.

Step 2: In this we compare all the characters sequentially from second last to the second character until a complete match or a mismatch occurs. If all the characters are matched, we get the desired pattern and this is notified and our algorithm moves to the next stage. If mismatch occurs it directly jumps to the stage 3.

Step 3: Here we shift the window according to the shift calculated in the preprocessing phase. This shift is obtained by taking the maximum of the shift provided by $brBc$ and $newBc$ and the window is shifted to the right according to that. All the three stages of the searching phase are repeated until the window is positioned beyond $n - m + 1$.

Table 2 shows a comparison (number of text character comparison) of SSABS, TVSBS and proposed algorithm for the text (T) and pattern (P) taken in FASTA format:

T=ATCTAACATCATAACCCGAGTTGGCATTGGCAGGAGAGCCAATCGATG
P = ATTGGCAG

Table 2. Comparison among SSABS, TVSBS and Proposed algorithm

ALGORITHM	SSABS	TVSBS	Proposed algorithm
ATTEMPT	15	8	7
COMPARISONS	30	23	17

3.3 Extension to Parameterized Matching

By using the concept of q -gram and RGF string [15], the proposed algorithm has been extended for parameterized string matching. A sequence of q -characters is being considered for calculating the bad character shift. Bad character shift in this algorithm is calculated for q -consecutive RGF encoded characters. During the searching process, q -consecutive characters of the text are RGF encoded online. Rest algorithm remains the same.

4 Experimental Results

We have implemented our proposed algorithm and existing algorithms on the same data set in C++, compiled with GCC 4.2.4 compilers on the Pentium 4, 2.14 GHz processor with 512 MB RAM, running ubuntu 10.04. We performed several experiment on different pattern length and various text sizes. Pattern and text are taken from file: <http://genome.crg.es/datasets/genomics96/seqs/DNASequences.fasta> over the alphabet {A, C, G, T}. Table 3 gives the number of comparison of the algorithms for varying pattern length. Table 4 gives the running time of the algorithms for varying pattern length. Table 5 gives the number of comparison of the algorithms for varying text length. Table 6 gives the running time of the algorithms for varying text length. Table 7 shows the behavior of our proposed algorithm on increasing the duplicity present in the code (for same pattern and text length).

Table 3. No. of comparisons \times 100000 of various algorithms
(For varying pattern length)

Pattern length	SSABS	TVSBS	Proposed Algorithm
4	118	160	82
8	62	112	58
12	100	62	38
16	64	66	34
20	78	60	20
24	120	66	24
36	40	60	18
48	56	38	16

Table 4. Running time in seconds of various algorithms
(For varying pattern length)

Pattern length	SSABS	TVSBS	Proposed Algorithm
4	.052	.068	.060
8	.038	.045	.040
12	.054	.032	.030
16	.042	.038	.028
20	.044	.028	.024
24	.054	.028	.024
36	.028	.040	.022
48	.036	.022	.020

Table 5. No. of comparisons \times 100000 of various algorithms
(For varying text length)

Text length (MB)	SSABS	TVSBS	Proposed Algorithm
0.5	0.5	0.4	0.2
1	0.8	0.5	0.3
2	1.8	1.2	0.5
3	2.8	1.8	0.8
4	3.6	2.2	1.0
5	4.5	2.8	1.2
6	5.5	3.5	1.5

Table 6. Running time in seconds of various algorithms
(For varying text length)

Text length (MB)	SSABS	TVSBS	Proposed Algorithm
0.5	0.05	0.040	0.030
1	0.07	0.050	0.050
2	0.14	0.008	0.008
3	0.18	0.012	0.010
4	0.26	0.016	0.014
5	0.03	0.020	0.018
6	0.36	0.022	0.020

Table 7. Effect on running time on increasing the duplicity present in the text
(For fixed pattern length, $q = 3$ and text size = 6 MB)

Parameterized Alphabet	Fixed alphabet	Time (s)
-	{A, C, G, T}	0.020
{A}	{C, G, T}	0.050
{A, C}	{G, T}	0.102
{A, C, G}	{T}	0.167
{A, C, G, T}	-	0.212

5 Conclusions

In this paper, we propose an algorithm which gives the best possible shift in the search phase and is faster than the previously known algorithms. The proposed

algorithm is compared with existing algorithm on the same data set to compare the relative performance. Tables 3 and 4 shows that on increasing the pattern length, our algorithm performs the best in terms of number of comparison as well as in the running time. Tables 5 and 6 shows that, on increasing the text length, our algorithm is still best among the others. Table 7 shows that on increasing the duplicity present in code (i.e. increasing the plagiarism), running time increase.

References

1. Roy, C.K., Cordy, J.R.: A survey on software clone detection research, Technical Report (2007)
2. Whale, G.: Software Metrics and Plagiarism Detection. *Journal of System Software* 13, 131–138 (1990)
3. Boyer, R.S., Moore, J.S.: A fast string searching algorithm. *Communication of ACM* 20, 762–772 (1977)
4. Amir, A., Navarro, G.: Parameterized Matching on Non-linear Structures. *Information Processing Letters (IPL)* 109(15), 864–867 (2009)
5. Sunday, D.M.: A very fast substring search algorithm. *Communication of ACM* 33, 132–142 (1990)
6. Berry, T., Ravindran, S.: A fast string matching algorithm and experimental results. In: Holub, J., Simánek, M. (eds.) *Proceedings of the Prague Stringology Club Workshop 1999*, Collaborative Report DC-99-05, Czech Technical University, Prague, Czech Republic, pp. 16–26 (2001)
7. Baker, B.S.: A program for identifying duplicated code. In: *Computing Science and Statistics: Proceedings of the 24th Symposium on the Interface*, vol. 24, pp. 49–57 (1992)
8. Sheik, S.S., Aggarwal, S.K., Poddar, A., Balakrishnan, N., Sekar, K.: A FAST pattern matching algorithm. *J. Chem. Inf. Comput. Sci.* 44, 1251–1256 (2004)
9. Horspool, R.N.: Practical fast searching in strings. *Software – Practice and Experience* 10, 501–506 (1980)
10. Raita, T.: Tuning the Boyer–Moore–Horspool string-searching algorithm. *Software – Practice Experience* 22, 879–884 (1992)
11. Abrahamson, K.: Generalized String Matching. *SIAM Journal on Computing* 16, 1039–1051 (1987)
12. Charras, C., Lecroq, T.: Handbook of Exact String matching algorithms, <http://www-igm.univ-mlv.fr/~lecroq/string/>
13. Thathoo, R., Virmani, A., Laxmi, S.S., Balakrishnan, N., Sekar, K.: TVSBS: A fast exact pattern matching algorithm for biological sequences. *Current Science* 91(1) (2006)
14. Baker, B.S.: Parameterized pattern matching by Boyer-Moore type algorithms. In: *Proceedings of the 6th ACM-SIAM Annual Symposium on Discrete Algorithms*, San Francisco, CA, pp. 541–550 (1995)
15. Salmela, L., Tarhio, J.: Fast Parameterized Matching with q-Grams. *Journal of Discrete Algorithms* 6(3), 408–419 (2008)
16. Prasad, R., Agarwal, S., Misra, S.: Parameterized String Matching Algorithms with Application to Molecular Biology. *Nigerian Journal of Technological Research*, 5 (2010)

Dynamic Software Maintenance Effort Estimation Modeling Using Neural Network, Rule Engine and Multi-regression Approach

Ruchi Shukla¹, Mukul Shukla^{2,3}, A.K. Misra⁴, T. Marwala⁵, and W.A. Clarke¹

¹ Department of Electrical & Electronic Engineering Science
University of Johannesburg, South Africa

² Department of Mechanical Engineering Technology
University of Johannesburg, South Africa

³ Department of Mechanical Engineering
Motilal Nehru National Institute of Technology, Allahabad, India

⁴ Department of Computer Science and Engineering
Motilal Nehru National Institute of Technology, Allahabad, India

⁵ Faculty of Engineering and Built Environment
University of Johannesburg, South Africa

{ruchishuklamtech,mukulshukla2k}@gmail.com, akm@mnunit.ac.in,
{tmarwala,willemc}@uj.ac.za

Abstract. The dynamic business environment of software projects typically involves a large number of technical, demographic and environmental variables. This coupled with imprecise data on human, management and dynamic factors makes the objective estimation of software development and maintenance effort a very challenging task. Currently, no single estimation model or tool has been able to coherently integrate and realistically address the above problems. This paper presents a multi-fold modeling approach using neural network, rule engine and multi-regression for dynamic software maintenance effort estimation. The system dynamics modeling tool developed using quantitative and qualitative inputs from real life projects is able to successfully simulate and validate the dynamic behavior of a software maintenance estimation system.

Keywords: Software maintenance, effort estimation, system dynamics, neural network, regression.

1 Introduction

In any dynamic business environment a large number of technical, demographic, environmental and social variables are typically involved [1],[2]. This situation is particularly valid in case of software projects. The dynamics of software projects coupled with lack of data on human, management and dynamic factors makes the objective estimation of dynamic software effort a very challenging task. Current estimation models and tools are still striving to integrate and address all of the above problems. The need is to develop a system dynamics model using quantitative and

qualitative inputs to realistically simulate system behavior for software maintenance estimation [3].

Dynamic simulation models can play a vital role to understand the software process evolution as compared to static simulation models. Dynamic project performance estimation aims at ascertaining a software maintenance project's performance in terms of expected schedule, effort or defect density, when the project conditions are dynamically changing during the program execution [4]. The dynamic simulation model of software projects helps to capture and model the impact of technological changes in management policies, team dynamics and organizational maturity level over the entire duration of the project. It objectively represents the behavioural observations of the software development process and models the change of system state variables continuously.

The dynamic software process simulation models (SPSMs) are used to understand, plan, manage and improve software development processes [5]. Caivano *et al.* presented a tool for dynamic effort estimation of an aged software system. Method characteristics like fine granularity and dynamic tuning permit the tool to quickly react to process dynamics [6]. Many researchers [7],[8],[9] have attempted to develop SPSMs to address different issues related to software projects. These dynamic models use number of parameters and functions to characterize the environment (project and organizational). However, in the absence of historical project databases and due to software projects uncertainty it becomes difficult to set values for these parameters and build a new simulation model to validate the relationships among project variables [10]. Further, customization of the published simulation models for an organization is often not so easy due to unavailability of sufficient control variables to represent the specific characteristics of various types of software projects. Choi and Bae presented a dynamic software process simulation modeling method based on COCOMO II. Their framework applied expert judgment technique to mitigate the unavailability of empirical data on the effects of a dynamic project environment. The developed model could handle unanticipated and uncontrolled creep requirements of the project [4]. Jorgensen and Ostvold analysed the reasons for occurrence of errors in software effort estimation. They found that the data collection approach, the role of the respondents, and the type of analysis had an important impact on the reasons given for estimation error [1].

Simulation of the dynamics of software development projects helps to investigate the impact of technological change of different management policies and maturity level of organizations over the whole project. Bhatt *et al.* collected outsourced software maintenance data (*unavailable in public domain*) and described a system dynamics model to maintain a software system [11]. However, they did not account for outliers and missing information, and gave only an empirical model to calculate the impact of the considered factors on software maintenance effort.

In the past system dynamics with simulation models have been applied successfully in the context of software engineering estimation. Models have been built that successfully predicted changes in project cost, staffing needs and schedules over time, as long as the proper initial values of project development are available to the estimator [12]. Calzolari *et al.* suggested how dynamic systems could be

successfully applied in the software maintenance context, to model the dynamic evolution of software maintenance effort [13]. Baldassarre *et al.* presented SPEED, a tool for implementing the dynamic calibration approach for renewal of legacy systems, where the current and well established approaches tend to fail. They applied the method of dynamic calibration for effort estimation also and conducted its experimental validation [14].

Recent approaches of software development and maintenance for dynamic systems have used hybrid or multiple methods which integrate quantitative and qualitative information. When quantitative models are unavailable, a correctly trained neural network (NN) can be used for modeling of non-linear and/or dynamic systems. The authors used the NN approach previously for modeling of software maintenance effort based on 14 *static* effort drivers [15]. However, NNs are unable to provide reasonable insight into model behavior as they are explicit rather than implicit in form. This difficulty of NN can be overcome using qualitative modeling or rule-based inference methods. For example, genetic algorithm can be used together with NNs to enhance reasoning capabilities [16]. Hence, in the present work a multi-fold modeling approach using neural network, rule engine and multi-regression has been adopted to precisely estimate the *dynamic* software maintenance effort.

2 Software Maintenance Effort Estimation Methodology

Based on a detailed study conducted [17],[18],[19],[20] and various respondents from industry (*Infosys, Syntel, Netlink and IBM*) on how practical software maintenance is carried out in the software industry, the following steps are proposed for software maintenance effort estimation:

1. *Identification of Cost Drivers*: Emphasizing on maintenance activities, the main maintenance cost drivers are identified and information is collected of project size (LOC, FP), maintenance productivity etc. Their respective values are obtained from industry and used as input parameters for effort estimation.
2. *Evaluate Complexity*: Performed by capturing information from a request or application based on the complexity of change for a particular file or program.
3. *Identification of Dynamic Issues*: Dynamic issues are identified carefully and categorized as dynamic factors and their respective complexities are calculated based on static values.
4. *Associating Complexity with Required Effort*.
5. *Determining Cause-Effect Relationship between Maintenance Effort and Various Effort Drivers*.

3 Dynamic Factors

A large number of variables are involved in a dynamic software business setup. Many of these factors are time dependent and change dynamically according to the ongoing changes. From a literature survey of software maintenance effort estimation it was

inferred that while calculating maintenance effort most of the researchers have not emphasized on the impact of various dynamic issues. Some of these issues include changing configuration, change in user requirements, volatility, turnover of project personnel, extension of project deadline and change to modern means of documentation. It is therefore necessary to bring these dynamic issues in a modeling approach to realistically simulate the behavior of the system dynamics. In this work, the following dynamic factors are included for dynamic software maintenance effort estimation [21]:

1. *Traceability factor* – As per IEEE standard 610, traceability is defined as a degree to which a relationship can be established between two or more products of development process. Traceability factor keeps an account of the dynamic changes in requirements, specifications, platform etc. While making any software, attempts are made to use tools to match the requirements, to design, to code, thus achieving a higher traceability factor. Higher the traceability factor, higher is the maintainer's productivity and lower is the maintenance effort [21].
2. *Usability factor* - Any software should satisfy all levels of requirements from the user's perspective. A properly updated user manual helps in minimizing the maintenance cost, increasing usability and maintenance productivity, thereby decreasing maintenance effort.
3. *Configuration management factor* - It has the following tasks: software change request (SCR) and approval procedures, code (and related work products) and change management tasks and software configuration, auditing managing and tracking software documentation, releases etc. If these are followed properly then accordingly it is rated dynamically. Higher the configuration management factor lower is the maintenance effort.
4. *Volatility factor* - This factor accounts for software systems which are known to have frequent modifications and up gradation (e.g. stock exchange, banking etc). Higher the volatility factor, higher is the maintenance effort [7].
5. *Turnover factor* - This factor helps in dynamically analysing the time frame in which a maintenance project is completed. It plays a vital role in medium to large sized projects [3].

4 Data Collection and Analysis

In the present work, responses were collected from software maintenance industry experts (including those from Infosys, Syntel, IBM and others) to generate own dataset by adopting a questionnaire based survey approach. The data was collected across professionals of different educational qualifications, job roles and working at various geographical locations across different companies. Further, these professionals had various levels of total software experience (minimum 3 years) as well as with maintenance activities (minimum 2 years) for software in different application domains.

Information was acquired from experts for rating of the 5 dynamic factors on a Likert scale (1 to 5) based on the static drivers' settings for different projects. Details of the dataset of 36 commercially outsourced SM projects based on 14 cost drivers

and effort are as follows [22]. The estimation was carried out by 6 experts each at Syntel, an ISO 9001 certified and SEI CMM level 5 application management and e-business solutions company. This data is based on a legacy insurance system project running on the IBM platform with a total system size of 0.138 million lines and 1275 number of programs.

The corresponding dynamic effort was also acquired from experts based on the available static effort and Likert rating of the 5 dynamic factors. A total of 50 questionnaires were forwarded and 11 responses received. Some of the missing information and superfluous information from the responses received were supplemented based on inputs and experience of similar projects. Data beyond $\pm 20\%$ of the upper and lower limits was also appended suitably.

5 Dynamic Effort Estimation

From a literature review and experts feedback the cause-effect relationship of the 8 most dominant static factors and the 5 dynamic factors (traceability factor - TF, configuration management factor - CMF, usability factor – UF, volatility factor – VF and turnover factor - TOF) was obtained, as presented in Table 1. For dynamic software maintenance effort estimation modeling, the effect of only these 8 top ranked significant (total % contribution to effort nearly 88%) static effort factors (N, D, B, M, C, L, F and E) was considered.

Table 1. Cause-effect relationship between the eight most significant static factors and the five dynamic factors

Effort Drivers	Code	Rank	Cause	Effect on				
				TF	CMF	UF	VF	TOF
No. of time zones the users that need support are spread across	N	1	↓	↑	↑	↑	↓	↓
Average no. of lines per program	D	2	↓	↑	---	↑	---	↓
% of the online programs to the total number of programs	B	3	↓	↑	↑	↑	↓	↓
Nature of SLA	M	4	↑	↑	↑	↑	↓	↓
Complexity of file system used	C	5	↔	↑	↑	↑	↓	↓
% of the update programs to the total number of programs	L	6	↓	↑	↑	↓	↓	↓
No. of database objects used by the system	F	7	↓	↑	↑	↑	↓	↓
No. of files (input, output, sort) used by the system	E	8	↓	↑	↑	↑	↓	---

where, ↓↑ indicates decreasing and increasing trend, respectively; --- indicates insignificant trend; and ↔ indicates medium trend)

The dynamic effort has been found using the following two approaches:

- 1) *Input from Experts* - Based on the rating of various dynamic factors on a scale of 1 to 5, and the static effort, the corresponding dynamic effort was acquired by experts as explained in Section 4.
- 2) *Rule Based System* – A Rule Engine has been developed and applied for the estimation of dynamic effort. From the information acquired by experts for complexity based rating of various dynamic factors their values have been found. The detailed steps of the proposed rule based system are explained below [3]:
 - i) At first, a schema for assignment of complexity to the 3 levels of the 8 most significant static effort factors (N, D, B, M, C, L, F and E) was designed, based on the feedback obtained from practicing software maintenance personnel. A ternary scheme was adopted to match the 3 levels of static factors. The finally assigned complexity values to different levels of 8 significant static factors are shown in Table 2.

Table 2. Assigned complexity to different levels of 8 significant static factors

Code	Static effort factors	Level 1		Level 2		Level 3	
		Value	Cmplx.	Value	Cmplx.	Value	Cmplx.
N	Number of time zones the users that need support are spread across	1	+1	3	0	5	-1
D	Average number of lines per program	750	+1	2250	0	3750	-1
B	Percentage of the online programs to the total number of programs	10%	+1	30%	0	50%	-1
M	Nature of SLA	1	-1	5	0	10	+1
C	Complexity of the file system being used	1	-1	2	+1	3	0
L	Percentage of the update programs to the total number of programs	7%	+1	30%	0	53%	-1
F	Number of database objects used by the system	5	+1	20	0	35	-1
E	Number of files (input, output, sort) used by the system	125	+1	375	0	625	-1

- ii) Using the cause-effect relationship of the 8 most dominant static factors and the 5 dynamic factors (Table 1) the complexity of the 5 dynamic factors was calculated as follows:

$$TF_complexity = N_complexity + D_complexity + B_complexity + M_complexity + C_complexity + L_complexity + F_complexity + E_complexity$$

$$UF_complexity = N_complexity + D_complexity + B_complexity + M_complexity + C_complexity - L_complexity + F_complexity + E_complexity$$

$$CMF_complexity = N_complexity + B_complexity + M_complexity + C_complexity + L_complexity + F_complexity + E_complexity$$

$$VF_complexity = -(N_complexity + B_complexity + M_complexity + C_complexity + L_complexity + F_complexity + E_complexity)$$

$$TOF_complexity = -(N_complexity + D_complexity + B_complexity + M_complexity + C_complexity + L_complexity)$$

- iii) Based on the complexity obtained in the above step the dynamic rating for the 5 dynamic factors was fixed on a scale of 1 to 5 as per the following schema:

If (dyn_factor)_complexity > -9 & < -4 then (dyn_factor) rating = 1

If (dyn_factor)_complexity > -5 & < -1 then (dyn_factor) rating = 2

If (dyn_factor)_complexity > -2 & < 2 then (dyn_factor) rating = 3

If (dyn_factor)_complexity > 1 & < 5 then (dyn_factor) rating = 4

If (dyn_factor)_complexity > 4 & < 9 then (dyn_factor) rating = 5

This schema was forwarded to various practicing software maintenance personnel and based on the feedback obtained from majority of them it was finalized.

- iv) The dynamic effort was eventually calculated using the following approach:

for i=1 to n (no_of_projects)

for j=1 to 5 (no_of_dyn_factors)

$$Dynamic_effort(i) = Static_effort(i) + \sum weight(j) * rating_dyn_factor(j) . \quad (1)$$

All the dynamic factors were assigned equal weights but their relative impact/ratings kept changing. The final results obtained from the above steps are shown in Table 3 (columns 7-11, 13).

Finally, a twofold modeling approach based on multi-linear regression and neural network techniques was adopted for dynamic effort estimation, as explained underneath in Sections 5.1 and 5.2, respectively. The results obtained from these

techniques were used for validation of the mean dynamic effort estimated by the experts and that calculated by the rule engine approach (column 13).

5.1 Regression Model for Dynamic Effort Estimation

Based on the 19 (14 static and 5 dynamic) factors along with the corresponding dynamic effort for different projects (as estimated by experts), multi-variable linear regression was carried out using the Minitab software, for dynamic software maintenance effort estimation. An ANOVA based multi-linear regression equation (Eq. 2) was obtained for estimation of dynamic software maintenance effort.

$$\begin{aligned}
 \text{Dynamic_Effort} = & 7.94 + 0.215*A + 0.021*B + 0.495*C + 0.0004*D + 0.001*E + \\
 & 0.035*F + 0.005*G + 0.063*H + 0.025*I + 0.031*J + \\
 & 0.052*K - 0.0009*L + 0.12*M + 0.319*N + \\
 & 0.158*TF + 0.856*UF - 0.586*CMF + 0.42*TOF . (2)
 \end{aligned}$$

5.1.1 Analysis and Validation of Results

The fifth dynamic factor (volatility factor - VF), though significant in a dynamically changing software maintenance project environment, was dropped from the regression equation, as it was highly correlated to another factor. Based on actual values of static and dynamic factors the dynamic effort was predicted using the regression equation (Eq. 2) as shown in column 14 of Table 3. It can be inferred from the % error values (column 15) between the estimated dynamic effort (using Rule Based System) and the regression model based dynamic effort that the regression model is able to successfully predict the dynamic effort with a high accuracy. The variation in error is of the order of +6.85 to -5.51% with an average % error of 0.002, validating the proposed approach and the reasonable goodness of fit of the model (square of coefficient of regression $R^2 = 0.912$; Fig. 1).

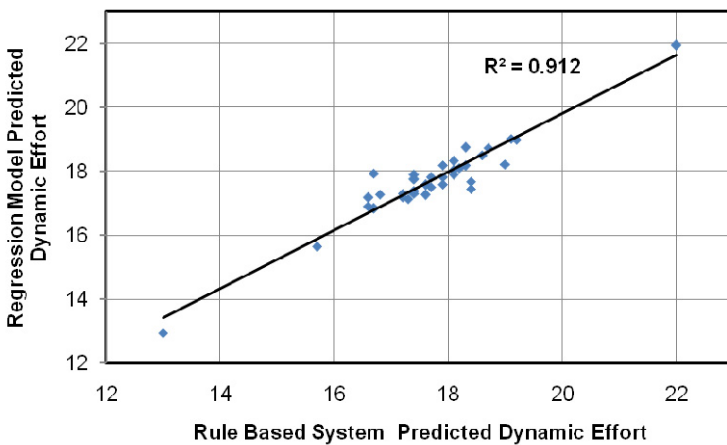


Fig. 1. Comparison of rule based system and regression model predicted dynamic effort

Table 3. Comparison of rule based system estimated and regression model predicted dynamic effort

Proj.	Static factors			Dynamic factors						Mean Static effort (Rao & Sarda [22])	Estimated Dynamic effort (Rule Based System) C13	Regression model predicted Dynamic effort (Eq. 2) C14	% Error (1-C13/C14) *100	
	A	B	C to L	M	N	TF	UF	CMF	VF					TOF
1	1	10		1	1	4	4	4	2	2	9.8	13	12.95	-0.39
2	1	10		5	3	4	4	4	2	2	14.1	17.3	17.14	-0.92
3	1	10		10	5	2	3	2	4	3	19.2	22	21.94	-0.27
4	1	10		10	5	4	4	3	3	3	14.5	17.9	17.59	-1.75
5	1	10		1	1	4	3	4	2	2	12.7	15.7	15.64	-0.40
6	1	10		5	3	2	2	3	3	3	15.1	17.7	17.51	-1.06
7	2	10		5	5	3	3	2	4	3	14.7	17.7	17.80	0.55
8	2	10		10	1	4	5	4	2	2	14.0	17.4	17.91	2.87
9	2	10		1	3	3	3	4	2	3	14.4	17.4	17.39	-0.06
10	2	10		10	3	4	3	3	3	2	14.2	17.2	17.19	-0.07
11	2	10		1	5	3	3	3	3	3	14.4	17.4	17.75	1.97
12	2	10		5	1	3	4	4	2	3	14.2	17.4	17.31	-0.55
13	1	30		1	3	2	2	2	4	4	13.9	16.7	16.83	0.77
14	1	30		5	5	3	3	3	3	4	15.1	18.3	18.74	2.36
15	1	30		10	1	4	3	4	2	2	13.6	16.6	16.90	1.75
16	1	30		5	1	2	3	2	4	3	14.0	16.8	17.27	2.72
17	1	30		10	3	4	3	4	2	2	15.1	18.1	18.03	-0.41
18	1	30		1	5	3	3	3	3	3	13.6	16.6	17.18	3.40
19	2	30		5	5	3	2	3	3	3	14.8	17.6	17.58	-0.09
20	2	30		10	1	4	4	4	2	2	15.0	18.2	18.10	-0.54
21	2	30		1	3	2	3	2	4	3	15.6	18.4	17.44	-5.51
22	2	30		10	3	2	3	2	4	3	15.1	17.9	18.19	1.59
23	2	30		1	5	3	2	3	3	3	15.6	18.4	17.68	-4.09
24	2	30		5	1	4	4	4	2	2	14.4	17.6	17.26	-1.98
25	1	50		5	3	3	2	3	3	4	15.1	18.1	17.89	-1.15
26	1	50		10	5	3	3	3	3	3	16.0	19	18.22	-4.30
27	1	50		1	1	2	3	2	4	4	15.6	18.6	18.51	-0.46
28	1	50		1	5	1	2	1	5	5	16.3	19.1	19.02	-0.42
29	1	50		5	1	3	3	3	3	2	14.4	17.2	17.31	0.65
30	1	50		10	3	4	4	4	2	3	15.3	18.7	18.73	0.14
31	2	50		1	1	1	1	2	4	4	14.8	17.2	17.19	-0.08
32	2	50		5	3	4	4	3	3	3	14.5	17.9	17.82	-0.47
33	2	50		10	5	3	2	3	3	3	16.4	19.2	18.99	-1.11
34	2	50		10	1	3	3	3	3	3	15.3	18.3	18.18	-0.65
35	2	50		1	3	2	3	2	4	3	15.3	18.1	18.32	1.19
36	2	50		5	5	3	2	3	3	3	16.3	16.7	17.93	6.85
Average % error												0.002		

5.2 NN Modeling for Dynamic Effort Estimation

The following NN modeling scheme has been adopted in this work. As compared to the NN based *static* effort estimation of [15], instead of 14 static factors, now a total of 19 (14 static and 5 dynamic) factors along with the corresponding *dynamic* effort for different projects (as estimated by experts) were used as input. Back-propagation NN modeling with incremental (instead of fixed) learning and Bayesian regularization training algorithm were used for better prediction of dynamic effort. The simplest possible architecture i.e. a 3 layer NN (19-19-1), with 19 neurons in the input layer, 19 neurons in the first hidden layer and 1 neuron for the single response (dynamic effort) in the third layer was fixed, as shown in Fig. 2.

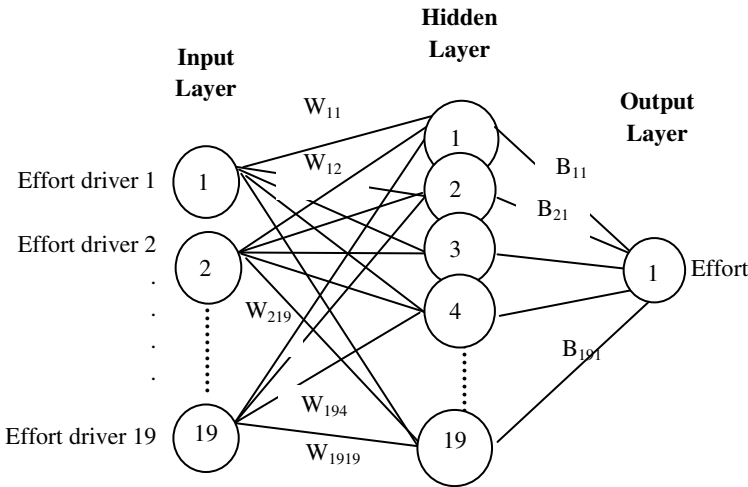


Fig. 2. NN architecture for dynamic effort estimation

Here, the log-sigmoid activation function in hidden layer and linear activation function in the output layer have been used. The number of hidden nodes was increased from 19 to 35 and the reduction in SSE observed. During trials, the minimum SSE varied randomly with increased hidden nodes. Hence, the simplest NN architecture with 1 hidden layer and minimum hidden neurons was chosen. The NN was similarly trained using 50% of input dataset and rest 25% each were used for validation and testing. Once properly trained, the NN successfully dealt with data of new projects and estimated the values of dynamic maintenance effort and % error.

5.2.1 Analysis and Validation of Results

Further, NN analysis was performed and training results on available data using a 19-19-1 architecture were obtained. The results were on expected lines as the test set error and the validation set error had similar characteristics and converged very fast. The sum of squared errors SSE for training, testing and validation were also fairly acceptable. A comparison of the NN output with expert estimated experimental values

of dynamic effort gives an error variation from -18.25% to 9.63%, 8.72% to -7.18% and 6.12% to -4.39% for training, testing and validation, respectively. The respective average error though is significantly smaller at 8.39%, 6.48% and 2.86%, respectively. (Interested readers can refer to [3] for details) Linear regression analysis between the actual network outputs (A) and the corresponding experimental data targets (T) was performed as shown in Fig. 3. However, the two outliers still showed larger simulation errors. Even a simplified NN architecture was able to successfully model the complex relationship between the 19 input variables and the single output variable. However, the results are not as accurate as is evident from the correlation coefficient ‘R’ value (around 0.85) for multiple runs of the code (Fig. 3). Further studies employing different NN architectures and input parameters as carried out in [15] can be attempted to improve the above results.

6 AI Based Dynamic Effort Estimation Tool (AIDEE)

Using Java Net Beans IDE 6.8 environment a user friendly, AI Based Dynamic Effort Estimation Tool (AIDEE) was also developed. A facility of integrating it with MATLAB 7 based NN code has also been provided for validating the empirical results with NN. Thus the tool has artificial intelligence based dynamic effort estimation capability. The tool estimates both the static and dynamic maintenance effort based on regression models and validates the available/entered values for respective projects for NN simulation (Fig. 3). Further details of the AIDEE tool are available in [3].

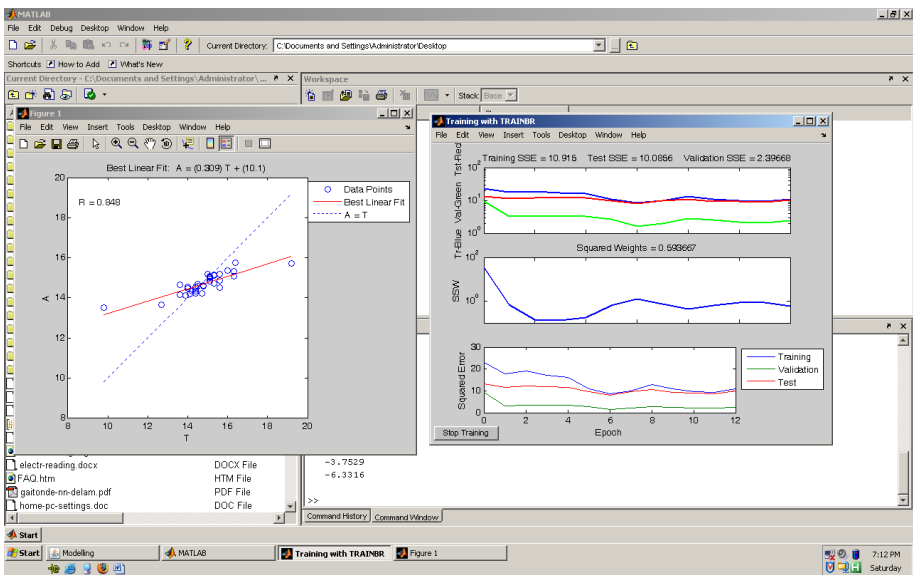


Fig. 3. Snapshot of NN modeling and regression plot of dynamic effort in AIDEE

7 Conclusions

In this work, for the first time, a set of 5 dynamic factors have been incorporated for dynamic software maintenance effort modeling. Based on rating of dynamic factors and the static effort, the corresponding dynamic effort data was acquired by software maintenance professionals. A rule based system has been proposed which was ratified by the industry experts to design a methodology to estimate dynamic maintenance effort. A regression based approach for empirical modeling of the estimated dynamic effort in software maintenance jobs is also presented. Alternately, the dynamic effort has also been estimated using the NN approach. A NN based estimation tool AIDEE is developed to successfully model the complex relationship of 19 different types of static and dynamic effort drivers with dynamic effort. The effectiveness of the three proposed approaches is duly emphasized from the closeness of obtained results. For future research, the present approach can be validated against a wider variety of real life software maintenance projects in different domains. This multi-approach is likely to help in accounting for the dynamic environment of modern day software maintenance projects and accurate estimation of software maintenance effort.

Acknowledgments. The financial assistance provided by the Faculty of Engineering and Built Environment, University of Johannesburg to RS is greatly acknowledged.

References

1. Jorgensen, M., Ostvold, K.M.: Reasons for Software Effort Estimation Error: Impact of Respondent Role, Information Collection Approach, and Data Analysis Method. *Trans. Softw. Eng.* 30(12), 993–1007 (2004)
2. Bhatt, P., Shroff, G., Misra, A.K.: Dynamics of Software Maintenance. *ACM SIGSOFT SEN* 29(5), 1–5 (2004)
3. Shukla, R.: Static and Dynamic Software Maintenance Effort Estimation: An Artificial Intelligence and Empirical Approach, PhD Thesis, MNNIT Allahabad, India (2011)
4. Choi, K.S., Bae, D.H.: Dynamic Project Performance Estimation by Combining Static Estimation Models with System Dynamics. *Inf. Softw. Tech.* 51(1), 162–172 (2009)
5. Donzelli, P., Iazeolla, G.: A Hybrid Software Process Simulation Model. *Softw. Proc. Improv. Pract.* 6(2), 97–109 (2001)
6. Caivano, D., Lanubile, F., Visaggio, G.: Software Renewal Process Comprehension Using Dynamic Effort Estimation. In: *Proceedings of the 17th IEEE International Conference on Software Maintenance*, Florence, Italy, pp. 209–218 (2001)
7. Pfahl, D., Lebsanft, K.: Using Simulation to Analyze the Impact of Software Requirements Volatility on Project Performance. *Inf. Softw. Tech.* 42(14), 1001–1008 (2000)
8. Mackulak, G., Collofello, J.: Stochastic Simulation of Risk Factor Potential Effects for Software Development Risk Management. *J. Syst. Softw.* 59(3), 247–257 (2001)
9. Ruiz, M., Ramos, I., Toro, M.: A Simplified Model of Software Project Dynamics. *J. Syst. Softw.* 59, 299–309 (2001)
10. Haberlein, T.: Common Structure in System Dynamics Models of Software Acquisition Projects. *Softw. Proc. Improv. Pract.* 9(2), 67–80 (2004)

11. Bhatt, P., Shroff, G., Anantram, C., Misra, A.K.: An Influence Model for Factors in Outsourced Software Maintenance. *J. Softw. Maint. Evol: Res. Pract.* 18(6), 385–423 (2006)
12. Hamid, T.K.A., Madnick, S.: *Software Project Dynamics: An Integrated Approach*. Prentice-Hall, Englewood Cliffs (1991)
13. Calzolari, F., Tonella, P., Antoniol, G.: Maintenance and Testing Effort Modeled by Linear and Nonlinear Dynamic Systems. *Inf. Softw. Tech.* 43(8), 477–486 (2001)
14. Baldassarre, M.T., Boffoli, N., Caivano, D., Visaggio, G.: SPEED: Software Project Effort Evaluator Based on Dynamic-Calibration. In: *Proceedings of the 22nd International Conference on Software Maintenance*, Philadelphia, pp. 272–273 (2006)
15. Shukla, R., Misra, A.K.: Estimating Software Maintenance Effort - A Neural Network Approach. In: *Proceedings of the 1st India Software Engineering Conference (ISEC)*, Hyderabad, pp. 107–112. ACM Digital Library (2008)
16. Shukla, K.K.: Neuro-Genetic Prediction of Software Development Effort. *Inf. Softw. Tech.* 42, 701–713 (2000)
17. Lucia, A.D., Pompella, E., Stefanucci, S.: Assessing the Maintenance Processes of a Software Organization: An Empirical Analysis of a Large Industrial Project. *J. Syst. Softw.* 65(2), 87–103 (2003)
18. IEEE Standard, ISO/IEC, 14764, *Software Engineering - Software Life Cycle Processes - Maintenance* (2006)
19. Hung, V.T. (2007), <http://cnx.org/content/m14719/latest>
20. Pigoski, T.M.: *Practical software maintenance*. John Wiley & Sons, Inc. (1997)
21. Shukla, R., Misra, A.K.: AI Based Framework for Dynamic Modeling of Software Maintenance Effort Estimation. In: *Proceedings of the International Conference on Computer and Automation Engineering (ICCAE)*, Bangkok, pp. 313–317 (2009)
22. Rao, B.S., Sarda, N.L.: Effort Drivers in Maintenance Outsourcing - An Experiment Using Taguchi's Methodology. In: *Proceedings of the 7th IEEE European Conference on Software Maintenance and Reengineering*, Benevento, Italy, pp. 1–10 (2003)

New Measures for Maintaining the Quality of Databases

Hendrik Decker*

Instituto Tecnológico de Informática, Universidad Politécnica de Valencia, Spain

Abstract. Integrity constraints are a means to model the quality of databases. Measures that size the amount of constraint violations are a means to monitor and maintain the quality of databases. We present and discuss new violation measures that refine and go beyond previous inconsistency measures. They serve to check updates for integrity preservation and to repair violations in an inconsistency-tolerant manner.

1 Introduction

The quality of databases can be identified with the satisfaction of the integrity constraints imposed on the data, and the lack of quality with their violation. In [8,11], we have seen how integrity constraints can be used to model, measure and monitor the quality of information stored in databases.

Building on that idea, we have shown in [10] how to identify, compute and measure cases and causes of integrity violations in order to control the quality of data in terms of their consistency. In [8,11,10], we have defined several measure-based methods which check updates for integrity preservation in an inconsistency-tolerant manner. Inconsistency-tolerant integrity checking (abbr. ITIC) means that only updates that do not increase a measured amount of inconsistency are acceptable, no matter if any constraint violation may already exist before the execution of a given update.

In this paper, we take the idea of using constraints and the measurement of their violation for controlling the quality and integrity of databases two steps further. Firstly, we present several new measures of integrity violation that enable a refined assessment of data quality. Secondly, we show how such measures can be used not only for ITIC, but also for inconsistency-tolerant integrity repair.

In Section 2, we define the formal framework of the remainder. In Section 3, we refine the axiomatization of violation measures developed in [8,11] and define several new measures that go beyond those defined in [10]. In Section 4, we show how the new violation measures can be used to maintain database integrity, by checking updates and repairing inconsistencies. In Section 5, we conclude.

* Partially supported by FEDER and the Spanish grants TIN2009-14460-C03 and TIN2010-17139.

2 The Framework

In 2.1, we outline some basic preliminaries. In 2.2 we recapitulate the notion of ‘cases’ from 8.1.1. In 2.3, we extend the notion of ‘causes’ from 10. Causes are instances of constraints that are useful for three objectives: simplified integrity checking, quantifying constraint violations and tolerating inconsistency. Causes are the data that are responsible for the violation of constraints, and are of similar use as cases. We use notations and terminology that are known from *datalog* 16 and first-order predicate logic.

2.1 Databases, Completions, Updates, Constraints

An *atom* is an expression of the form $p(t_1, \dots, t_n)$, where p is a predicate of arity n ($n \geq 0$); the t_i are either constants or variables. A *literal* is either an atom A or a negated atom $\sim A$. A *database clause* is a universally closed formula of the form $A \leftarrow B$, where the *head* A is an atom and the *body* B is a possibly empty conjunction of literals. If B is empty, A is called a *fact*. If B is not empty, $A \leftarrow B$ is called a *rule*. As is well-known, rules are useful for defining view predicates, as well as for enabling deductive and abductive reasoning capabilities in databases. A *database* is a finite set of database clauses. A database is *definite* if there is no negated atom in the body of any of its clauses.

Let us assume a universal Herbrand base \mathcal{H} and a universal set \mathcal{N} of constants, represented w.l.o.g. by natural numbers, that underlie each database.

Let $comp(D)$ denote the well-known completion of D 6. It essentially consists of the if-and-only-if completions (in short, completions) of all predicates in the language of D . For a predicate p , let p_D denote the completion of p in D .

Definition 1. Let D be a database, p a predicate, n the arity of p , x_1, \dots, x_n the \forall -quantified variables in p_D , and θ a substitution of x_1, \dots, x_n .

- a) For $A = p(x_1, \dots, x_n)\theta$, the *completion* of A in D , be obtained by applying θ to p_D , and be denoted by A_D .
- b) Let $comp(D) = \{A_D \mid A \in \mathcal{H}\}$.
- c) Let $if(D)$ and *only-if*(D) be obtained by replacing \leftrightarrow in each $A_D \in comp(D)$ by \leftarrow and, resp., \rightarrow .
- d) Let $iff(D) = if(D) \cup only\text{-}if(D)$. Let the usual equality axioms of $comp(D)$ be associated by default also to $iff(D)$.

Clearly, $if(D)$ is equivalent to the set of all ground instances of clauses in D . Moreover, $comp(D)$, $comp(D)$ and $iff(D)$ clearly have the same logical consequences. However, the characterization of causes in 2.2.3 by subsets of $iff(D)$ is more precise than it could be if subsets of $comp(D)$ were used instead.

We may use ‘;’ instead of ‘,’ to delimit elements of sets since ‘,’ is also used to denote conjunction in the body of clauses. Symbols \models , \Rightarrow and \Leftrightarrow denote logical consequence (i.e., truth in all Herbrand models), meta-implication and, resp., meta-equivalence. By overloading, we use $=$ as identity predicate, assignment in substitutions, or meta-level equality; \neq is the negation of $=$.

An *update* is a finite set of database clauses to be inserted or deleted. For an update U of a database D , we denote the database in which all inserts in U are added to D and all deletes in U are removed from D , by D^U . An *update request* in a database D is a sentence R that is requested to become *true* by updating D . An update U *satisfies* an update request R in D if $D^U(R) = \text{true}$. View updating is a well-known special kind of satisfying update requests. From now on, repairs are treated as updates, and repairing as satisfying specific update requests.

An *integrity constraint* (in short, *constraint*) is a sentence which can always be represented by a *denial clause*, i.e., a universally closed formula of the form $\leftarrow B$, where the body B is a conjunction of literals that asserts what *should not* hold in any state of the database. If the original specification of a constraint by a sentence I expresses what *should* hold, then a denial form of I can be obtained by an equivalence-preserving re-writing of $\leftarrow \sim I$ as proposed, e.g., in [7], that results in a denial the predicates of which are defined by clauses to be added to the database. An *integrity theory* is a finite set of constraints.

From now on, let the symbols D , IC , I , U and adornments thereof always stand for a database, an integrity theory, a constraint and, resp., an update, each of which is assumed, as usual, to be range-restricted [7].

For each sentence F , and in particular for each integrity constraint, we write $D(F) = \text{true}$ (resp., $D(F) = \text{false}$) if F evaluates to *true* (resp., *false*) in D . Similarly, we write $D(IC) = \text{true}$ (resp., $D(IC) = \text{false}$) if each constraint in IC is satisfied in D (resp., at least one constraint in IC is violated in D). Let $\text{vioCon}(D, IC)$ denote the set of violated constraints in D .

2.2 Cases

For each constraint I , a *case* of I is an instance of I obtained by substituting the variables in I with (not necessarily ground) terms of the underlying language. This definition of cases is a simpler version of a more encompassing variant in [12], where cases have been defined for the purpose of inconsistency-tolerant integrity checking of constraints in a more general syntax.

Reasoning with cases of I instead of I itself lowers the cost of integrity maintenance, since, the more variables in I are instantiated with ground values, the easier the evaluation of the so-obtained case tends to be. Also, to know which particular cases of a constraint are violated may be useful for repairing, since it turns out to be easier, in general, to identify and eliminate the causes of integrity violation if the violated cases are made explicit.

Let $\text{Cas}(IC)$ denote the set of all ground cases of each $I \in IC$. Further, let $\text{vioCon}(D, IC) = \{I \mid I \in IC, D(I) = \text{false}\}$, i.e., the set of all constraints in IC that are violated in D , and $\text{vioCas}(D, IC) = \{C \mid C \in \text{Cas}(IC), D(C) = \text{false}\}$, i.e., the set of all violated ground cases of IC in D .

The usefulness of cases for simplified integrity checking is well-known and well documented, e.g. in [5]. The use of $\text{vioCon}(D, IC)$ and $\text{vioCas}(D, IC)$ for measuring the inconsistency of (D, IC) is addressed in Section 3, their use for inconsistency-tolerant integrity maintenance in Section 4.

2.3 Causes

As in [10], we are going to define a ‘cause’ of the violation of a constraint $I = \leftarrow B$ in a database D as a minimal explanation of why I is violated in D , i.e., why the existential closure $\exists B$ of B is *true* in D . However, the definition of causes below is much more general than its predecessor in [10]. The latter applied only to definite databases and integrity theories without negation in the body of denials. In this paper, that restriction is dropped. A significant consequence of this generalization is that the logic to be used for reasoning with cases and causes of constraints must comply with the non-monotonicity of negation in databases.

In Section 3, causes are used for measuring inconsistency, and in Section 4 for measure-based inconsistency-tolerant integrity maintenance.

Definition 2. Let D be a database and $I = \leftarrow B$ an integrity constraint such that $D(\exists B) = \text{true}$. A subset E of $\text{iff}(D)$ is called a *cause of the violation of I in D* if $E \models \exists B$, and for each proper subset E' of E , $E' \not\models \exists B$.

We also say that E is a *cause of $\exists B$ in D* if E is a cause of the violation of $\leftarrow B$ in D . Moreover, we say that, for an integrity theory IC , E is a *cause of the violation of IC in D* if E is a cause of the violation of a denial form of the conjunction of all constraints in IC .

Clearly, E is a cause of the violation of each denial form of the conjunction of all $I \in IC$ if and only if E is a cause of the violation of some $I \in IC$ and there is no cause E' of any constraint in IC such that $E' \subsetneq E$.

For easy reading, we represent elements of *only-iff*(D) in a simplified form, if possible, in the subsequent examples of causes. Simplifications are obtained by replacing ground equations with their truth values and by common equivalence-preserving rewritings for the composition of subformulas with *true* or *false*.

Example 1.

a) Let $D = \{p \leftarrow q, \sim r; q\}$.

The only cause of the violation of $\leftarrow p$ in D is $D \cup \{\sim r\}$.

b) Let $D = \{p(x) \leftarrow q(x), r(x); q(1); q(2); r(2); s(1); s(2)\}$. The only cause of the violation of $\leftarrow s(x), \sim p(x)$ in D is $\{s(1), p(1) \rightarrow q(1) \wedge r(1), \sim r(1)\}$.

c) Let $D = \{p \leftarrow q(1, x); q(2, y) \leftarrow r(y); r(1)\}$.

The only cause of $\sim p$ in D is $\{p \rightarrow \exists x q(1, x)\} \cup \{\sim q(1, i) \mid i \in \mathcal{N}\}$.

d) Let $D = \{p \leftarrow q(x, x); q(x, y) \leftarrow r(x), s(y); r(1); s(2)\}$. Each cause of $\sim p$ in D contains $\{p \rightarrow \exists x q(x, x)\} \cup \{q(i, i) \rightarrow r(i) \wedge s(i) \mid i \in \mathcal{N}\} \cup \{\sim r(2), \sim s(1)\}$ and, for each $j > 2$ in \mathcal{N} , either $\sim r(j)$ or $\sim s(j)$, and nothing else.

e) Let $D = \{p(x) \leftarrow r(x); r(1)\}$ and $I = \exists x(r(x) \wedge \sim p(x))$. A denial form of I is $\leftarrow \text{vio}$, where *vio* is defined by $\{\text{vio} \leftarrow \sim q; q \leftarrow r(x), \sim p(x)\}$, where q is a fresh 0-ary predicate. Thus, the causes of the violation of I in D are the causes of *vio* in $D' = D \cup \{\text{vio} \leftarrow \sim q; q \leftarrow r(x), \sim p(x)\}$. Thus, for each $\mathcal{K} \subseteq \mathcal{N}$ such that $1 \in \mathcal{K}$, $\{\text{vio} \leftarrow \sim q\} \cup \{p(i) \leftarrow r(i) \mid i \in \mathcal{K}\} \cup \{q \rightarrow \exists x(r(x) \wedge \sim p(x))\} \cup \{\sim r(i) \mid i \notin \mathcal{K}\}$ is a cause of *vio* in D' .

The following example shows that causes are not compositional, i.e., the causes of the violation of an integrity theory IC are not necessarily the union of the causes of the violation of the constraints in IC .

Example 2. Let $D = \{r(1, 1); s(1)\}$, $I_1 = \leftarrow r(x, x)$, $I_2 = \leftarrow r(x, y), s(y)$ and $IC = \{I_1, I_2\}$. The only cause of the violation of IC in D is $\{r(1, 1)\}$, which is a proper subset of the single cause D of the violation of I_2 in D .

Let $\text{vioCau}(D, IC)$ be the set of all causes of the violation of IC in D .

Clearly, $\text{vioCau}(D, IC)$ is analogous to vioCas as defined in [2.2]. While vioCas locates inconsistency by focusing on violated constraints, vioCau localizes inconsistency on the data that are responsible for integrity violations.

3 Violation Measures

Violation measures are a special kind of inconsistency measures [15]. Their purpose is to size the amount of integrity violation in databases. In [3.1], we semi-formally sketch our approach of violation measures. In [3.2], we define this concept formally. In [3.3], we first recapitulate some measures already defined in [8,11,10], and then introduce and discuss some new ones. In [3.4], we discuss some properties that are commonly associated to measures and investigate to which extent they apply to violation measures.

3.1 Conceptualizing Violation Measures

In [3.2], we are going to define an abstract concept of violation measures as a mapping from pairs (D, IC) to a set \mathbb{M} that is structured by a partial order \preceq with an infimum o , a distance δ and an addition \oplus with neutral element o .

The partial order \preceq allows to compare the amount of inconsistency in two pairs of databases and integrity theories, and in particular in consecutive states (D, IC) and (D^U, IC) . With the distance δ , the difference, i.e., the increase or decrease of inconsistency between D and D^U , can be sized. The addition \oplus allows to state a standard metric property for δ , and o is, at a time, the smallest element of \preceq and the neutral element of \oplus .

Thus, there are various measurable criteria according to which an update U can be checked while tolerating extant inconsistencies, e.g., if U does not increase the amount of inconsistency, or if D^U does not trespass a certain threshold of inconsistency, or if any increase of inconsistency brought about by U is negligible.

In traditional measure theory [1], a measure μ maps elements of a measure space \mathbb{S} (typically, a set of sets) to a metric space $(\mathbb{M}, \preceq, \delta)$ (typically, $\mathbb{M} = \mathbb{R}_0^+$, i.e., the non-negative real numbers, often with an additional greatest element ∞ , $\preceq = \leq$, and $\delta = | - |$, i.e., the absolute difference). For $S \in \mathbb{S}$, $\mu(S)$ usually tells how ‘big’ S is. Standard properties are that μ is *positive definite*, i.e., $\mu(S) = 0 \Leftrightarrow S = \emptyset$, μ is *additive*, i.e., $\mu(S \cup S') = \mu(S) + \mu(S')$, for disjoint sets $S, S' \in \mathbb{S}$, and μ is *monotone*, i.e., if $S \subseteq S'$, then $\mu(S) \leq \mu(S')$. The distance δ maps $\mathbb{M} \times \mathbb{M}$ to itself, for determining the difference between measured entities.

Similarly, for assessing inconsistency in databases, a violation measure ν as defined in 3.2 maps pairs (D, IC) to a metric space that, like \mathbb{R}_0^+ , has a partial order and an addition with neutral element. The purpose of $\nu(D, IC)$ is to size the amount of inconsistency in (D, IC) , e.g., by counting or comparing sets of cases or causes of constraint violations.

3.2 Formalizing Violation Measures

Definitions 3 and 4 below specialize the traditional concepts of metric spaces and measures, since they are made up for databases and integrity violations. Yet, in a sense, these definitions also generalize the traditional concepts, since they allow to size and compare different amounts of inconsistency without necessarily quantifying them numerically. For example, with $\mathbb{M} = 2^{Cas(IC)}$ (powerset of $Cas(IC)$ as defined in 2.2.2), $\preceq = \subseteq$ (subset), $\delta = \ominus$ (symmetric set difference), $\oplus = \cup$ (set union) and $o = \emptyset$ (empty set), it is possible to measure the inconsistency of (D, IC) by sizing $vioCas(D, IC)$.

Definition 3. A structure $(\mathbb{M}, \preceq, \delta, \oplus, o)$ is called a *metric space for integrity violation* (in short, a *metric space*) if (\mathbb{M}, \oplus) is a commutative semi-group with neutral element o , \preceq is a partial order on \mathbb{M} with infimum o , and δ is a distance on \mathbb{M} . More precisely, for each $m, m', m'' \in \mathbb{M}$, the following properties (1)–(4) hold for \preceq , (5)–(8) for \oplus , and (9)–(11) for δ .

$$m \preceq m \quad (\text{reflexivity}) \tag{1}$$

$$m \preceq m', m' \preceq m \Rightarrow m = m' \quad (\text{antisymmetry}) \tag{2}$$

$$m \preceq m', m' \preceq m'' \Rightarrow m \preceq m'' \quad (\text{transitivity}) \tag{3}$$

$$o \preceq m \quad (\text{infimum}) \tag{4}$$

$$m \oplus (m' \oplus m'') = (m \oplus m') \oplus m'' \quad (\text{associativity}) \tag{5}$$

$$m \oplus m' = m' \oplus m \quad (\text{commutativity}) \tag{6}$$

$$m \oplus o = m \quad (\text{neutrality}) \tag{7}$$

$$m \preceq m \oplus m' \quad (\oplus\text{-monotonicity}) \tag{8}$$

$$\delta(m, m') = \delta(m', m) \quad (\text{symmetry}) \tag{9}$$

$$\delta(m, m) = o \quad (\text{identity}) \tag{10}$$

$$\delta(m, m') \preceq \delta(m, m'') \oplus \delta(m'', m') \quad (\text{triangle inequality}) \tag{11}$$

Let $m \prec m'$ ($m \succ m'$) denote that $m \preceq m'$ (resp., $m' \preceq m$) and $m \neq m'$.

Example 3. $(\mathbb{N}_0, \leq, | - |, +, 0)$ is a metric space for integrity violation, where \mathbb{N}_0 is the set of non-negative integers. In this space, $vioCon(D, IC)$, $vioCas(D, IC)$ or $vioCau(D, IC)$ can be counted and compared. As already indicated, these three sets may also be sized and compared in the metric spaces $(2^X, \subseteq, \ominus, \cup, \emptyset)$, where X stands for IC , $Cas(IC)$ or $iff(D)$, respectively.

The following definition generically characterizes violation measures, whose ranges are metric spaces such as those in Example 3.

Definition 4. A *violation measure* (in short, a *measure*) is a function ν that maps pairs (D, IC) to a metric space $(\mathbb{M}, \preceq, \delta, \oplus, o)$ for integrity violation.

3.3 New Violation Measures

We continue with examples of violation measures that are going to accompany us throughout the remainder of the paper. Preliminary versions of some of them have already been presented in [8,11,10]. However, the axiomatization of those previous versions is more shallow than in 3.2. Also the study of various properties of violation measures in 3.4 is very scant in the cited predecessors.

Example 4. A coarse measure β is defined by $\beta(D, IC) = D(IC)$ with the binary metric space $(\{true, false\}, \preceq, \tau, \wedge, true)$, where \preceq and τ are defined by stipulating $true \prec false$ (i.e., satisfaction means lower inconsistency than violation), and, resp., $\tau(v, v') = true$ if $v = v'$, else $\tau(v, v') = false$, for $v, v' \in \{true, false\}$, i.e., the value of $\tau(v, v')$ is the truth value of the logical equivalence $v \leftrightarrow v'$. Clearly, β and its metric space reflect the classical logic distinction that a set of formulas is either consistent or inconsistent, without any further differentiation of different degrees of inconsistency. The meaning of τ is that each consistent pair (D, IC) is equally good, and each inconsistent pair (D, IC) is equally bad.

Example 5. Two measures ι and $|\iota|$ that are quite straightforward are characterized by comparing and, resp., counting the set of violated constraints in the integrity theory of the database schema. The formal definition of these measures is given by the equations $\iota(D, IC) = vioCon(IC, D)$ and $|\iota|(D, IC) = |\iota(D, IC)|$, where $|\cdot|$ is the cardinality operator, with metric spaces $(2^{IC}, \subseteq, \ominus, \cup, \emptyset)$ and, resp., $(\mathbb{N}_0^+, \leq, |-|, +, 0)$.

Example 6. Two measures that are more fine-grained than those in Example 5 are given by $\zeta(D, IC) = vioCas(IC, D)$ and $|\zeta|(D, IC) = |\zeta(D, IC)|$, with metric spaces $(2^{Cas(IC)}, \subseteq, \ominus, \cup, \emptyset)$ and, resp., $(\mathbb{N}_0^+, \leq, |-|, +, 0)$.

Example 7. Similar to cases, cause-based measures can be defined by the equations $\kappa(D, IC) = vioCau(IC, D)$ and $|\kappa|(D, IC) = |\kappa(D, IC)|$, with metric spaces $(2^{iff(D)}, \subseteq, \ominus, \cup, \emptyset)$ and, resp., again $(\mathbb{N}_0^+, \leq, |-|, +, 0)$. Specific differences between measures based on cases (as in Example 6) and measures based on causes (as in this example) have been discussed in [10].

Other violation measures are discussed in [8,11] among them two variants of an inconsistency measure in [14] that are applicable to databases with integrity constraints, based on quasi-classical models [2]. Essentially, both violation measures size the set of conflicting atoms in (D, IC) , i.e., atoms A such that both A and $\sim A$ are *true* in the minimal quasi-classical model of $D \cup IC$. Hence, their metric spaces are $(2^{\mathcal{H}^*}, \subseteq, \ominus, \cup, \emptyset)$ where \mathcal{H}^* is the union of \mathcal{H} and $\{\sim A \mid A \in \mathcal{H}\}$, and, resp., $(\mathbb{N}_0^+, \leq, |-|, +, 0)$.

Some more new measures are going to be identified in 3.4.1 and 4.1.

3.4 Properties of Violation Measures

As opposed to classical measure theory and previous work on inconsistency measures, Definition 4 does not require any axiomatic property of measures, such as positive definiteness, additivity or monotony. These usually are required for each traditional measure μ , as already mentioned in 3.1. We are going to look at such properties, and argue that positive definiteness is not cogent, and both additivity and monotony are invalid for many databases.

In 3.4.1, we discuss the standard axiom of positive definiteness of measures, including some weakenings thereof. In 3.4.2, we show the standard axiom of additivity of measures is invalid for violation measures. In 3.4.3, also the standard axiom of monotonicity of measures is dismissed for violation measures, and some valuable weakenings thereof are proposed.

3.4.1 Positive Definiteness

For traditional measures μ , positive definiteness means that $\mu(S) = 0$ if and only if $S = \emptyset$, for each $S \in \mathbb{S}$. For violation measures ν , that takes the form

$$\nu(D, IC) = o \Leftrightarrow D(IC) = true \quad (\text{positive definiteness}) \quad (12)$$

for each pair (D, IC) .

A property corresponding to (12) is postulated for inconsistency measures in 13. However, we are going to argue that (12) is not cogent for violation measures, and that even two possible weakenings of (12) are not persuasive enough as sine-qua-non requirements.

At first, (12) may seem to be most plausible as an axiom for any reasonable inconsistency measure, since it assigns the lowest possible inconsistency value o precisely to those databases that totally satisfy all of their constraints. In fact, it is easy to show the following result.

Theorem 1. Each of the measures $\beta, \iota, |\iota|, \zeta, |\zeta|, \kappa, |\kappa|$ fulfills (12).

Thus, in particular $|\zeta|$, which counts the number of violated ground cases, complies with (12). Now, let the measure ζ' be defined by the following modification of $|\zeta|$: $\zeta'(D, IC) = 0$ if $|\zeta|(D, IC) \in \{0, 1\}$ else $\zeta'(D, IC) = |\zeta|(D, IC)$. Thus, ζ' considers each inconsistency that consists of just a single violated ground case as insignificant. Hence, ζ' does not obey (12) but can be, depending on the application, a very reasonable violation measure that tolerates negligible amounts of inconsistency.

Even the weakening

$$D(IC) = true \Rightarrow \nu(D, IC) = o \quad (13)$$

of (12) is not a cogent requirement for all reasonable violation measures, as witnessed by the measure σ , as defined below. It takes a differentiated stance with regard to integrity satisfaction and violation, by distinguishing between satisfaction, satisfiability and violation of constraints, similar to 19 17.

The measure σ be defined by incrementing a count of ‘problematic’ ground cases of constraints by 1 for each ground case that is satisfiable but not a theorem

of the completion of the given database, and by 2 for each ground case that is violated. Hence, by the definitions of integrity satisfaction and violation in [17], there are pairs (D, IC) such that IC is satisfied in D but $\sigma(D, IC) > 0$.

Another measure ϵ that does not respect (I3) can be imagined as follows, for databases with constraints of the form $I = \leftarrow p(x), x > th$, where $p(x)$ is a relation defined by some aggregation of values in the database, meaning that I is violated if $p(x)$ holds for some x that trespasses a certain threshold th . Now, suppose that ϵ assigns a minimal non-zero value to (D, IC) whenever I is still satisfied in D but $D(p(th)) = true$, so as to indicate that I is at risk of becoming violated. Hence, there are pairs (D, IC) such that $\nu = \epsilon$ contradicts (I3).

Also the requirement

$$\nu(D, \emptyset) = o \tag{14}$$

which weakens (I3) even further, is not indispensable, although analogons of (I4) are standard in the literature on classical measures and inconsistency measures. In fact, it is easy to imagine a measure that assigns a minimal non-zero value of inconsistency to a database without integrity constraints. That value can then be interpreted as a warning that there is a non-negligible likelihood of inconsistency in a database where no constraints are imposed, be it out of neglect, or for trading off consistency for efficiency, or due to any other reason.

So, in the end, only the rather bland property $\nu(\emptyset, \emptyset) = o$ remains as a weakening of (I2) that should be required from violation measures.

3.4.2 Additivity

For traditional measures μ , additivity means $\mu(S \cup S') = \mu(S) + \mu(S')$, for each pair of disjoint sets $S, S' \in \mathbb{S}$. For violation measures ν , additivity takes the form

$$\nu(D \cup D', IC \cup IC') = \nu(D, IC) \oplus \nu(D', IC') \quad (\text{additivity}) \tag{15}$$

for each $(D, IC), (D', IC')$ such that D and D' as well as IC and IC' are disjoint.

Additivity is standard for traditional measures. However, (I5) is invalid for violation measures, as shown by the following example.

Example 8. Let $D = \{p\}, IC = \emptyset, D' = \emptyset, IC' = \{\leftarrow p\}$. Clearly, $D(IC) = true$ and $D'(IC') = true$, thus $|\zeta|(D, IC) + |\zeta|(D', IC') = 0$, but $|\zeta|(D \cup D', IC \cup IC') = 1$.

3.4.3 Monotony

For traditional measures μ , monotony means $S \subseteq S' \Rightarrow \mu(S) \preceq \mu(S')$, for each pair of sets $S, S' \in \mathbb{S}$. For violation measures ν , monotony takes the form

$$D \subseteq D', IC \subseteq IC' \Rightarrow \nu(D, IC) \preceq \nu(D', IC') \quad (\nu\text{-monotonicity}) \tag{16}$$

for each pair of pairs $(D, IC), (D', IC')$.

A property corresponding to (I6) is postulated for inconsistency measures in [13]. For *definite* databases and integrity theories (i.e., the bodies of clauses do not contain any negative literal), it is easy to show the following result.

Theorem 2. For each pair of definite databases D, D' and each pair of definite integrity theories IC, IC' , each of the measures $\beta, \iota, |\iota|, \zeta, |\zeta|, \kappa, |\kappa|$ fulfills (16).

However, due to the non-monotonicity of negation in the body of clauses, (16) is not valid for non-definite databases or non-definite integrity theories, as shown by Example 9, in which the foreign key constraint $\forall x(q(x, y) \rightarrow \exists z s(x, z))$ on the x -column of q referencing the x -column of s is rewritten into denial form (we ignore the primary key constraint on the x -column of s since it is not relevant).

Example 9. Let $D = \{p(x) \leftarrow q(x, y), \sim r(x); r(x) \leftarrow s(x, z); q(1, 2); s(2, 1)\}$ and $IC = \{\leftarrow p(x)\}$. Clearly, $D(IC) = false$ and $|\zeta|(D, IC) = 1$. For $D' = D \cup \{s(1, 1)\}$ and $IC' = IC$, we have $D'(IC') = true$, hence $|\zeta|(D', IC') = 0$.

Now, we are going to look at two weakenings of (16) that hold also for non-definite databases and integrity theories. In fact, (16) is already a weakening of (15), since it is easily shown that (15) and (8) entail (16). Hence, any valid weakening of (16) can also be understood as a valid weakening of (15).

The first weakening requires that the inconsistency in databases that violate integrity is never lower than the inconsistency in databases that satisfy integrity. Formally, for each pair of pairs $(D, IC), (D, IC')$, the following property is asked to hold.

$$D(IC) = true, D'(IC') = false \Rightarrow \mu(D, IC) \preceq \mu(D', IC') \tag{17}$$

It is easy to show the following result.

Theorem 3. Each of the measures $\beta, \iota, |\iota|, \zeta, |\zeta|, \kappa, |\kappa|$ fulfills (17).

A property that is slightly stronger than (17) has been postulated in [8,11]. It is obtained by replacing \preceq in (17) by \prec . It also holds for all measures from Subsection 3.3. Yet, similar to (12), it does not hold for measures ζ', σ and ϵ , as defined in 3.4.1, while (17) does hold for those measures.

The second weakening of (16) has been postulated in [10]. It requires that, for each D , the values of ν grow monotonically with growing integrity theories.

$$IC \subseteq IC' \Rightarrow \nu(D, IC) \preceq \nu(D, IC') \tag{18}$$

Since (18) weakens (16), the following result is entailed by Theorem 3 for $\beta, \iota, |\iota|, \zeta, |\zeta|, \kappa, |\kappa|$, and can be shown also for ζ', σ, ϵ .

Theorem 4. Each of the measures $\beta, \iota, |\iota|, \zeta, |\zeta|, \kappa, |\kappa|, \zeta', \sigma, \epsilon$ fulfills (18).

4 Integrity Maintenance

To maintain integrity, constraint violations should be prevented or repaired. For prevention, a common approach is to check if updated preserve integrity. For repairing, methods described, e.g., in [20] may be used. However, it may be impractical or unfeasible to avoid inconsistency, or to repair all violated constraints at once. Thus, an inconsistency-tolerant approach to integrity maintenance is

needed. As we are going to see, that can be achieved by violation measures. In fact, even in the presence of persisting inconsistency, the use of measures can prevent the increase of inconsistency across updates. Measures also are useful for controlling that the amount of inconsistency never exceeds given thresholds.

In [4.1](#), we revisit measure-based inconsistency-tolerant integrity checking (abbr. ITIC). Also, we show how inconsistency can be confined by assigning weights to violated cases of constraints, which goes beyond the measures seen so far. Moreover, we show how to generalize measure-based ITIC by allowing for certain increases of inconsistency that are bounded by some thresholds. In [4.2](#), we outline how measure-based inconsistency-tolerant integrity checking can be used also for making repairing inconsistency-tolerant.

4.1 Measure-Based Inconsistency-Tolerant Integrity Checking

[Definition 5](#), below, characterizes integrity checking methods that may accept updates if there is no increase of inconsistency, no matter if there is any extant constraint violation or not. It abstractly captures measure-based ITIC methods as black boxes, of which nothing but their i/o interface is observable. More precisely, each method \mathcal{M} is described as a mapping from triples (D, IC, U) to $\{ok, ko\}$. Intuitively, *ok* means that U does not increase the amount of measured inconsistency, and *ko* that it may.

Definition 5. (*Inconsistency-tolerant Integrity Checking*, abbr. *ITIC*)

An *integrity checking method* maps triples (D, IC, U) to $\{ok, ko\}$. For a measure (ν, \preceq) , a method \mathcal{M} is called *sound* (*complete*) for ν -based ITIC if, for each (D, IC, U) , [\(19\)](#) (resp., [\(20\)](#)) holds.

$$\mathcal{M}(D, IC, U) = ok \Rightarrow \nu(D^U, IC) \preceq \nu(D, IC) \quad (19)$$

$$\nu(D^U, IC) \preceq \nu(D, IC) \Rightarrow \mathcal{M}(D, IC, U) = ok \quad (20)$$

Each \mathcal{M} that is sound for ν -based ITIC is also called a ν -based method.

Intuitively, [\(19\)](#) says: \mathcal{M} is sound if, whenever it outputs *ok*, the amount of violation of IC in D as measured by ν is not increased by U . Conversely, [\(20\)](#) says: \mathcal{M} is complete if it outputs *ok* whenever the update U that is checked by \mathcal{M} does not increase the amount of integrity violation.

As opposed to ITIC, traditional integrity checking (abbr. TIC) imposes the *total integrity* requirement. That is, TIC additionally requires $D(IC) = true$ in the premises of [\(19\)](#) and [\(20\)](#). The measure used in TIC is β (cf. [Example 4](#)). Since ITIC is defined not just for β but for any violation measure ν , and since TIC is not applicable if $D(IC) = false$, while ITIC is, [Definition 5](#) generalizes TIC. Moreover, the definition of ITIC in [\[12\]](#) is equivalent to [Definition 5](#) for $\nu = \zeta$. Hence, the latter also generalizes ITIC as defined in [\[12\]](#).

In [\[12\]](#), we have shown that the total integrity requirement is dispensable for most TIC approaches. Similar to corresponding proofs in [\[12\]](#), it can be shown that not all, but most TIC methods, including built-in integrity checks in common DBMSs, are ν -based, for each $\nu \in \{\iota, |\iota|, \zeta, |\zeta|, \kappa, |\kappa|\}$. The following results are easily shown by applying the definitions.

Theorem 5. If a method \mathcal{M} is ν -based, then it is $|\nu|$ -based, for each $\nu \in \{\iota, \zeta, \kappa\}$. If \mathcal{M} is κ -based, then it is ζ -based. If \mathcal{M} is ζ -based, then it is ι -based. The converse of none of these implications holds.

Example 10 below, illustrates how the measures $|\iota|, |\zeta|, |\kappa|$ that count violated constraints, cases or causes thereof can be generalized by assigning weight factors to the counted entities. Such weights are useful for modeling application-specific degrees of violated integrity. A simple variant of such an assignment comes into effect whenever ‘soft’ constraints that *ought to* be satisfied are distinguished from ‘hard’ constraints that *must* be satisfied.

Example 10. Let lr and hr be two predicates that model a low, resp., high risk. Further, $I_1 = \leftarrow lr(x)$, $I_2 = \leftarrow hr(x)$, be a soft, resp., hard constraint for protecting against low and, resp., high risks, where lr and hr are defined by $lr(x) \leftarrow p(y,z)$, $x = y + z$, $x > th$, $z \geq y$ and $hr(x) \leftarrow p(y,z)$, $x = y + z$, $x > th$, $y > z$, resp., where th is a threshold value that should not be exceeded. and $p(8, 3)$ be the only cause of integrity violation in some database D . Now, for each $\nu \in \{\iota, \zeta, \kappa\}$, no ν -based method would accept the update $U = \{delete\ p(8, 3), insert\ p(3, 8)\}$, although the high risk provoked by $p(8, 3)$ is diminished to the low risk produced by $p(3, 8)$. However, measures that assign weights to cases of I_2 that are higher than those of I_1 can avoid that problem. For instance, consider the measure ω that counts the numbers n_1 and n_2 of violated cases of I_1 and, resp., I_2 in D , and assigns $f_1 n_1 + f_2 n_2$ to $(D, \{I_1, I_2\})$, where $0 < f_1 < f_2$. Clearly, each ω -based method will accept U . In fact, for $\nu \in \{|\iota|, |\zeta|, |\kappa|\}$, also each ν -based method would accept U , but it is easy to imagine a slightly more elaborated update U' such that ν -based methods would not accept U' but ω -based methods would.

4.2 Repairs

Roughly, repairing means to compute and execute an update in order to eliminate integrity violation. Thus, each repair can be identified with an update.

In 4.2.1, we formalize repairs and illustrate them by examples. In 4.2.2, we outline how to compute repairs.

4.2.1 Formalizing Repairs

In 12, we have distinguished *total* and *partial* repairs. The former eliminate all inconsistencies, the latter only some. Partial repairs tolerate inconsistency, since violated constraints may persist, as illustrated by Example 11.

Example 11. Let $D = \{p(a, b, c), p(b, b, c), p(c, b, c), q(a, c), q(c, b), q(c, c)\}$ and $IC = \{\leftarrow p(x, y, z), \sim q(x, z); \leftarrow q(x, x)\}$. Clearly, the violated cases of IC in D are $\leftarrow p(b, b, c), \sim q(b, c)$ and $\leftarrow q(c, c)$. Each of the updates $U_1 = \{insert\ q(b, c)\}$ and $U_2 = \{delete\ p(b, b, c)\}$ is a partial repair of (D, IC) , since both fix the violation of $\{\leftarrow p(b, b, c), \sim q(b, c)\}$ in D . Similarly, $U_3 = \{delete\ q(c, c)\}$ is a partial repair that fixes the violation of $\{\leftarrow q(c, c)\}$ in D .

Sadly, partial repairs may cause new violations, as shown in Example 12.

Example 12. Consider again Example 11. As opposed to U_1 and U_2 , U_3 causes a new violation: $\leftarrow p(c, b, c), \sim q(c, c)$ is satisfied in D but not in D^{U_3} . Thus, the partial repair $U_4 = \{delete\ q(c, c); delete\ p(c, b, c)\}$ is needed to eliminate the violation of $\leftarrow q(c, c)$ in D without causing any violation that did not exist before executing the partial repair.

Definition 6, below, generalizes the definition of partial repairs by requiring that each repair must decrease the measured amount of integrity violation.

Definition 6. (*Repair*) Let D be a database, IC an integrity theory such that $D(IC) = false$, ν a violation measure and U an update.

- a) U is said to *preserve integrity wrt. ν* if $\nu(D^U, IC) \preceq \nu(D, IC)$ holds.
- b) For a proper subset S of $Cas(IC)$ such that $D(S) = false$ and $D^U(S) = true$, U is called a *partial repair* of (D, IC) .
- c) U is called a *ν -based repair* of (D, IC) if $\nu(D^U, IC) \prec \nu(D, IC)$ holds. If, additionally, $D^U(IC) = false$, U is also called a *ν -based patch* of (D, IC) . Else, if $D^U(IC) = true$, U is called a *total repair* of (D, IC) .

Definition 6 could be slightly modified by replacing all occurrence of conditions $D^U(IC) = false$ and $D^U(IC) = true$ by $\nu(D, IC) \succ o$ and $\nu(D, IC) = o$, respectively. For each $\nu \in \{\iota, |\iota|, \zeta, |\zeta|, \kappa, |\kappa|\}$, that replacement yields a definition that is equivalent to Definition 6. Moreover, it is easy to show the following.

Theorem 6. For each pair (D, IC) and each $\nu \in \{\iota, |\iota|, \zeta, |\zeta|\}$, each ν -based patch of (D, IC) is a partial repair of (D, IC) .

Note that the converse of Theorem 6 does not hold, as seen in Example 12. Theorem 6 also does not hold for $\nu \in \{\kappa, |\kappa|\}$, since the violation of some case C may have n causes, $n > 0$, in some database D , and a repair U may just eliminate one of the causes that violate C . Then, for $\nu \in \{\kappa, |\kappa|\}$, $\nu(D^U, IC) \prec \nu(D, IC)$, i.e., U a ν -based patch but not a partial repair of (D, IC) since $vioCas(D, IC) = vioCas(D^U, IC)$, hence $D(S) = D^U(S) = false$.

In the literature, repairs usually are required to be total and, in some sense, minimal. Mostly, subset-minimality is opted for, but several other notions of minimality exist [4, 18]. Note that Definition 6 does not involve any notion of minimality. However, each repair in Example 11 is subset-minimal.

Unpleasant side effects of repairs such as U_3 can be avoided by checking if a given partial repair is a patch with any convenient measure-based method, as expressed in the following result. It follows from Definitions 5 and 6.

Theorem 7. For each tuple (D, IC) , each partial repair U of (D, IC) , each measure ν and each ν -based method \mathcal{M} , U is a ν -based patch if $\mathcal{M}(D, IC, U) = ok$.

4.2.2 Computing Repairs

Repairs can be computed by update methods, defined as follows.

Definition 7. An *update method* is an algorithm that, for each database D and each update request R , computes candidate updates U_1, \dots, U_n ($n \geq 0$) such that $D^{U_i}(R) = true$ ($1 \leq i \leq n$). For a measure ν , an update method \mathcal{UM} is *integrity-preserving wrt. ν* if each U_i computed by \mathcal{UM} preserves integrity wrt. ν .

Integrity-preserving update methods can be used to compute patches and repairs wrt. any measure ν , as shown in [12] for the special case of $\nu = \zeta$. Theorem 8 below generalizes that result.

For an update request R in a database D , several update methods in the literature work in two phases. First, a candidate update U such that $D^U(R) = true$ is computed. Then, U is checked for integrity preservation by some TIC method. If that check is positive, U is accepted. Else, U is rejected and another candidate update, if any, is computed and checked. Hence, Theorem 8, below, follows from Definition 7 and Theorem 7.

Theorem 8. For each measure ν , each update method that uses ν -based ITIC to check its computed candidate updates is integrity-preserving wrt. ν .

Example 13 shows what can go wrong if an update method that is not integrity-preserving is used.

Example 13. Let $D = \{q(x) \leftarrow r(x), s(x); p(a, a)\}$, R the view update request to insert $q(a)$, and $IC = \{\leftarrow p(x, x); \leftarrow p(a, y), q(y)\}$. To satisfy R , most update methods compute the candidate update $U = \{insert\ r(a); insert\ s(a)\}$. To check if U preserves integrity, most methods compute the simplification $\leftarrow p(a, a)$ of the second constraint in IC . For avoiding a possibly expensive disk access for evaluating the simplified case $\leftarrow p(a, a)$ of $\leftarrow p(a, y), q(y)$, TIC methods that are not inconsistency-tolerant may use the invalid premise that $D(IC) = true$, by reasoning as follows. The constraint $\leftarrow p(x, x)$ in IC is not affected by U and subsumes $\leftarrow p(a, a)$; hence, IC remains satisfied in D^U . Thus, such methods wrongly conclude that U preserves integrity, since the case $\leftarrow p(a, y), q(y)$ is satisfied in D but violated in D^U . By contrast, each ITIC method rejects U , so that $U' = U \cup \{delete\ p(a, a)\}$ can be computed for satisfying R . Clearly, U' preserves integrity, and even removes the violated case $\leftarrow p(a, a)$.

The following example illustrates a general approach of how patches and total repairs can be computed by update methods off the shelf.

Example 14. Let $S = \{\leftarrow B_1, \dots, \leftarrow B_n\}$ ($n \geq 0$) be a set of cases of constraints in an integrity theory IC of a database D . An integrity-preserving repair of (D, S) (which is total if $S = IC$) can be computed by each integrity-preserving update method, simply by running the update request $\sim vio_S$, where vio_S be defined by the clauses $vio_S \leftarrow B_i$ ($1 \leq i \leq n$).

So far, we have said nothing about computing any measure that may be used in integrity-preserving update methods. In fact, computing measures $\iota, |\iota|, \zeta, |\zeta|$

corresponds to the cost of searching SLDNF trees rooted at constraint denials, which can be exceedingly costly. The same correspondence holds for computing κ and $|\kappa|$ in databases and integrity theories without negation in the body of clauses. If negation may occur, the cost can even be higher, as evidenced in [9].

Fortunately, none of these measures needs to be computed explicitly. Instead of computing $\nu(D^U, IC)$ and $\nu(D^U, IC)$ entirely, it suffices to compute a superset approximation of the increment $\delta(\nu(D, IC), \nu(D^U, IC))$, as many TIC methods do, for $\nu = \zeta$. As attested by such methods, approximating the increment of inconsistency in consecutive states is significantly less costly than checking the inconsistency of entire databases. Moreover, for two integrity-preserving partial repair candidates U, U' of IC in D that do not repair the same set of violations, U is preferable to U' if $\delta(\nu(D, IC), \nu(D^{U'}, IC)) \prec \delta(\nu(D, IC), \nu(D^U, IC))$, since U eliminates more inconsistency from D than U' .

5 Conclusion

In theory, database quality can be achieved by either preventing or eliminating the violation of integrity constraints. In practice, however, integrity violation cannot always be prevented, and a total elimination of all violations often is infeasible. Thus, integrity maintenance must be inconsistency-tolerant.

In this paper, we have generalized the concept of inconsistency-tolerant integrity checking and repairing in [12]. We have axiomatized measures that determine the amount of violation in given databases with associated integrity theories. Using such measures, each update can be checked and accepted if it does not increase the measured violation. Similarly, each repair is acceptable if it decreases the measured violation.

Future work includes an application of the concept of measure-based inconsistency tolerance for computing answers that have integrity in databases with violated constraints, and the use of measure-based ITIC for concurrent transactions in distributed and replicated databases.

References

1. Bauer, H.: Maß- und Integrationstheorie, 2nd edn. De Gruyter (1992)
2. Besnard, P., Hunter, A.: Quasi-Classical Logic: Non-trivializable Classical Reasoning from Inconsistent Information. In: Froidevaux, C., Kohlas, J. (eds.) ECSQARU 1995. LNCS, vol. 946, pp. 44–51. Springer, Heidelberg (1995)
3. Ceri, S., Cochrane, R., Widom, J.: Practical Applications of Triggers and Constraints: Success and Lingering Issues. In: Proc. 26th VLDB, pp. 254–262. Morgan Kaufmann (2000)
4. Chomicki, J.: Consistent Query Answering: Five Easy Pieces. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 1–17. Springer, Heidelberg (2006)
5. Christiansen, H., Martinenghi, D.: On simplification of database integrity constraints. *Fundam. Inform.* 71(4), 371–417 (2006)
6. Clark, K.: Negation as Failure. In: Gallaire, H., Minker, J. (eds.) *Logic and Data Bases*, pp. 293–322. Plenum Press (1978)

7. Decker, H.: The Range Form of Databases and Queries or: How to Avoid Floundering. In: Proc. 5th ÖGAI. Informatik-Fachberichte, vol. 208, pp. 114–123. Springer (1989)
8. Decker, H.: Quantifying the Quality of Stored Data by Measuring their Integrity. In: Proc. DIWT 2009, Workshop SMM, pp. 823–828. IEEE (2009)
9. Decker, H.: Answers That Have Integrity. In: Schewe, K.-D., Thalheim, B. (eds.) SDKB 2010. LNCS, vol. 6834, pp. 54–72. Springer, Heidelberg (2011)
10. Decker, H.: Causes of the Violation of Integrity Constraints for Supporting the Quality of Databases. In: Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., Apduhan, B.O. (eds.) ICCSA 2011, Part V. LNCS, vol. 6786, pp. 283–292. Springer, Heidelberg (2011)
11. Decker, H., Martinenghi, D.: Modeling, Measuring and Monitoring the Quality of Information. In: Heuser, C.A., Pernul, G. (eds.) ER 2009. LNCS, vol. 5833, pp. 212–221. Springer, Heidelberg (2009)
12. Decker, H., Martinenghi, D.: Inconsistency-tolerant Integrity Checking. *TKDE* 23(2), 218–234 (2011)
13. Grant, J., Hunter, A.: Measuring the Good and the Bad in Inconsistent Information. In: Proc. 22nd IJCAI, pp. 2632–2637 (2011)
14. Hunter, A.: Measuring Inconsistency in Knowledge via Quasi-Classical Models. In: Proc. 18th AAAI & 14th IAAI, pp. 68–73 (2002)
15. Hunter, A., Konieczny, S.: Approaches to Measuring Inconsistent Information. In: Bertossi, L., Hunter, A., Schaub, T. (eds.) Inconsistency Tolerance. LNCS, vol. 3300, pp. 191–236. Springer, Heidelberg (2005)
16. Ramakrishnan, R., Gehrke, J.: Database Management Systems. McGraw-Hill (2003)
17. Sadri, F., Kowalski, R.: A theorem-proving approach to database integrity. In: Foundations of Deductive Databases and Logic Programming, pp. 313–362. Morgan Kaufmann (1988)
18. ten Cate, B., Fontaine, G., Kolaitis, P.: On the Data Complexity of Consistent Query Answering. To appear in Proc. 15th ICDT. LNCS, Springer (2012)
19. Vardi, M.: On the integrity of databases with incomplete information. In: Proc. 5th PODS, pp. 252–266. ACM Press (1986)
20. Wijsen, J.: Database repairing using updates. *ACM Trans. Database Syst.* 30(3), 722–768 (2005)

A New Way to Determine External Quality of ERP Software

Ali Orhan Aydin

Department of Computer Engineering
Gelisim University, Istanbul, Turkey
aliorhanaydin@gmail.com

Abstract. Today many production systems plan to use Enterprise Resource Planning (ERP) software to gain competitive advantage. ERP promises in improving efficiency of business processes. However, inappropriate software selection results in a very complex managerial problem which is implementation of ERP software. To reduce the relevant risk, ERP software purchasers need to determine conformance of the software to their requirements. This study aims to define the requirement levels of external quality characteristics and provide a guide to production systems to evaluate ERP software in a systematic manner. In the frame of this reference, a way of evaluating external quality of ERP software is put forward to reduce the risk taken before purchasing it.

Keywords: External Software Quality, Enterprise Resource Planning.

1 Introduction

Change in the market conditions increase competition among production systems since 1970s [1]. Shorter product lifetimes, high quality requirements, demanding customers and availability of diverse alternatives are significant factors that affect current market conditions. These competitive market conditions caused production systems to seek solutions to survive [2].

By using Information Technologies (IT), companies aim to get competitive advantage. In the beginning, organizations tried to utilize applications of Material Requirement Planning (MRP), Manufacturing Resources Planning (MRP II), Distribution Resources Planning (DRP), Capacity Requirements Planning (CRP) and Computer Integrated Manufacturing (CIM). The main objective is to use their resources more effectively and increase competition strength [3].

At the end of the 1980s organizations seek solutions to integrate these types of information systems to manage their flexible systems, supply customized product demand, control product complexity and plan resources more effectively by the use of IT [4]. Integration is considered as a key factor for getting advantage and Enterprise Resource Planning (ERP) begin to be used in organizations in early 1990s [3, 5, 6].

As stated by Davenport [7] ERP systems promise to restructure business processes of organizations, because they intend to solve problems caused by the lack of coordination between applications and business processes [8]. Moreover, those enterprise systems integrate all of the information that flow organization wide [9].

Today, ERP software and ERP software implementation market became nearly 50 billion EURO (€) [8]. Day after day, ERP software market grows and many organizations want to benefit from this software. On the other hand, after spending a lot of time and money some organizations state that they are unable to utilize its benefits that they intend to get by implementing ERP software [7, 9]. As stated by Hong and Kim [10], at least the three quarters of ERP projects are judged to be a failure.

Managers emphasize that ERP implementation projects are the most difficult system development projects [11]). Especially, ERP applications change procedures and processes of organizations into a software system. Due to complexity, time and work-force requirements, after implementation it nearly becomes impossible to rollback [8].

Due to the risks, before purchasing ERP software, it is necessary to determine if the ERP software bears on the ability to satisfy stated or implied needs of its customers by the use of Software Quality (SQ) models [12]. As stated in ISO 25000 [13] there is no general software classification system and the importance of quality characteristics for each type of software depends on the type of software. Requirement levels of software quality characteristics needs to be determined to evaluate particular type of software product, by the use of the Software Quality models. As an example, for web applications studies on determining software quality are performed by Calero and Olsina [15, 16]. Likewise, such a study is necessary for ERP software.

In this study, we determine requirement levels of software quality characteristics according to users' view of quality for ERP software. This set of quality characteristics and their requirement levels can be used as a checklist to evaluate if ERP software bears on the ability to satisfy stated or implied needs by ERP software purchasers. By the use of this checklist, it is possible to reduce the risk related to ERP software.

The remainder of this paper is organized as follows. Section 2 gives an overview of the associated literature on external software quality characteristics. Transactional backbone of ERP systems is reviewed in Section 3. In Section 4, a checklist including requirement levels of ERP software according to the user view is given and some formulas are introduced for the purpose of evaluating particular ERP software. In the last section of the paper, summary and conclusion are presented.

2 External Software Quality and Its Characteristics

The term quality is defined by Crosby [14] as “conformance of requirements”. Feigenbaum [17] puts forward another definition of quality as total composite characteristics of a product or service to meet the expectations of customer while Juran [18] states the phrase “fitness for use”. Although the quality definitions give guidance on definition of software quality, software quality can be described from five different perspectives [19]. These perspectives are transcendental view, user view, manufacturing view, product view and value based view. When these perspectives are evaluated a general definition for software quality can be described as follows: a total composite of characteristics of software product which bears on the ability to meet the stated or implied needs [13].

The definition of Software Quality shows that characteristics and requirements are the most important factors. Characteristics contribute to fulfilment of requirements

and software quality arises from characteristics which are appropriate to the requirements [20]. The relationship between these factors and software quality is given in Figure 1.



Fig. 1. Relationship between characteristics, requirements and software product [18]

One of the earliest studies that tried to provide a framework for Software Quality and its characteristics was proposed by McCall et.al. [21]. Another study is put forward to constitute a set of factors that affect Software Quality by Boehm et.al. [22]. Bowen et.al. [23] are proposed with a larger number of characteristics after few years. As seen below, these early studies try to constitute software quality model by providing a set of characteristics.

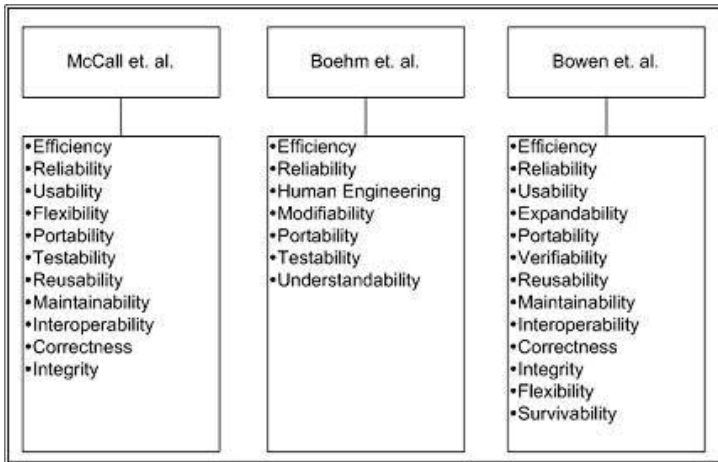


Fig. 2. Early software quality models

There are a lot of subsequent attempts to provide a software quality model [24-28]. Although, these sets of quality models including early models seem to cover the same identical characteristics, the definitions of these characteristics are different. ISO 25000 [12] software quality model is developed to establish international agreement and it is developed further with four parts by ISO [13, 29-31]. The international standard covers six quality characteristics which are shown in Figure 3 [13].

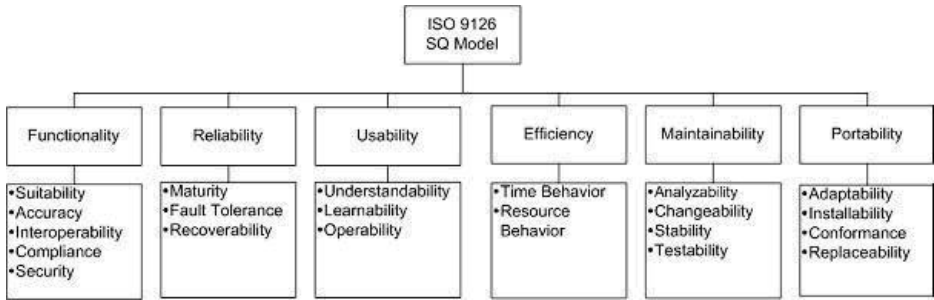


Fig. 3. ISO 25000 Software Quality Model [13]

Although international standard consolidated many different views of quality for software, there are still some other views that are not included into ISO 25000 and some of them cover more characteristics. In his model, Dromey [32] puts forward one more software quality characteristic, reusability. The other most significant attempt for constituting a set of characteristics is put forward by Software Engineering Committee [33]. Their model includes two more characteristics. These additional characteristics and sub-characteristics are given in Figure 4.

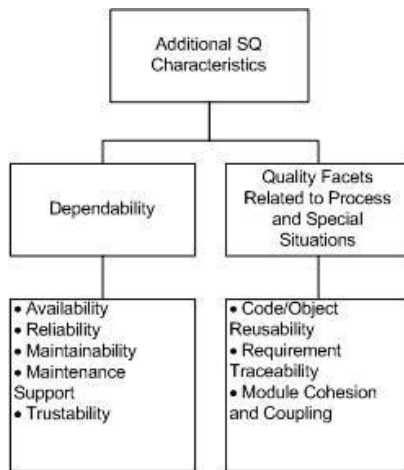


Fig. 4. Additional software quality characteristics [47, 48]

As it can be understood from these, there are some objection against international standard. However, debate on this issue is settled; since, there is no additional study after this standard. Therefore, we used software quality model of the standard.

As it is obviously seen in the definitions of quality, meeting the requirements of users is stressed. Likewise, Software Quality is also highly related to conforming to the requirements of end-users [34]. Moreover, proposed study aims to provide a guide

to ERP purchasers (i.e. end-users) to evaluate software products before implementing them. Therefore, External Software Quality Characteristics need to be explained.

As it is stated in the standard the relationship between software quality and its characteristics depends on the type of the user. The end-users are concerned with functionality, reliability, usability and efficiency characteristics [29]. These characteristics are called as External Software Quality Characteristics.

Definitions of these characteristics and sub-characteristics are clearly identified in the standard [29]. Since the study is based on external metrics, descriptions on characteristics are given in Table 1 and descriptions on sub-characteristics are given in Table 2.

Table 1. Descriptions of External Software Quality Characteristics [29]

Characteristics	Description
Functionality	The capability of the software to provide functions which meet stated and implied needs when the software is used under specified conditions.
Reliability	The capability of the software to maintain the level of performance of the system when used under specified conditions.
Usability	The capability of the software to be understood, learned, used and liked by the user, when used under specified conditions.
Efficiency	The capability of the software to provide the required performance relative to the amount of resources used, under stated conditions.

Table 2. Descriptions of External Software Quality Sub-Characteristics [29]

Characteristics	Sub-Characteristics	Description
Functionality	Suitability	Attribute of software that bears on the presence and appropriateness of a set of functions for specified tasks.
	Accuracy	Attributes of software that bear on the provision of right or agreed results or effects.
	Interoperability	Attributes of software that bear on its ability to interact with specified systems.
	Compliance	Attributes of software that make the software adhere to application-related standards or conventions or regulations in laws and similar prescriptions.
	Security	Attributes of software that bears on its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data.

Table 2. (continued)

Reliability	Maturity	Attributes of software that bear on the frequency of failure by faults in the software.
	Fault Tolerance	Attributes of software that bear on its ability to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.
	Recoverability	Attributes of software that bear on the capability to re-establish its level of performance and recover the data directly affected in case of a failure and on the time and effort needed for it.
Usability	Understandability	Attributes of software that bear on the users' effort for recognizing the logical concept and its applicability.
	Learnability	Attributes of software that bear on the users' effort for learning its application.
	Operability	Attributes of software that bear on the users' effort for operation and operation control.
Efficiency	Time Behaviour	Attributes of software that bear on response and processing times and on throughput rates in performing its function.
	Resource Behaviour	Attributes of software that bear on the amount of resources used and the duration of such use in performing its function.

3 Enterprise Resource Planning Systems

In this section of the study literature review on ERP systems is given; since, it is aimed to weight (i.e. determine the requirement level) each of the External Software Quality Characteristics and Sub-Characteristics to evaluate ERP. ERP systems usually cover a technical infrastructure, transactional backbone and advanced applications [35]. ERP systems can roughly be described as software that integrates distributed applications of finance, human resources, production, sales, purchase, supply and distribution [36].

Stated or implied expectations of production systems to utilize ERP systems can be found by elaborating on the chronological development processes of these systems. By this approach user expectations can be better understood.

It dates back to 1960s that the manufacturing systems first discovered Material Requirement Planning (MRP) [36]. MRP was the most effective tool for improving operations by calculating material requirements and requirement periods at that time

[37]. In the beginning of 1980s change in the market conditions caused manufacturing firms to seek innovative techniques. By adding new procedures to MRP, Manufacturing Resources Planning (MRP II) developed. MRP II try to integrate MRP and some other functional operation areas like marketing and finance [38]. Moreover; MRP II does not only cover MRP but also plans capacity [39].

After MRP II, Distribution Requirement Planning (DRP) and Computer Integrated Manufacturing (CIM) emerge. By using DRP, it becomes possible to plan and manage distribution channels and product deliveries [40]. CIM covers the applications of integration of the manufacturing processes and technical functions [3]. A few years after CIM and DRP, a new way of planning all of the resources of any organization Enterprise Resource Planning is born [35].

Explicit reason for evolution of ERP systems is to integrate former applications [6]. On the other hand, it must be noted that this is not the only reason that gives birth to ERP systems. In the last century small corporations changed into modern and global enterprises. Moreover, in the last three decades market conditions changed so quickly that those enterprises face hard competition [2]. Shortening of life cycle of products and rapid increase in the product diversity are the most significant factors all of which are shown to put enterprises under a big pressure [3].

Because of these conditions, production systems try to find a systemic idea involving all aspects of resource planning. They start to seek a way to restructure their processes flexibly. Most important factors for seeking this kind of integration is to enable production systems to deliver higher variety of products at lower cost, supply customized products, develop new production strategies focusing on individual customers and plan all of the resources more effectively [4].

It seemed possible to use IT in an integrated way to move towards the creation of appropriate infrastructure [41]. Under these circumstances the idea of Enterprise Resource Planning (ERP), which is able to plan all of the resources of organizations in an integrated way, is born. The main point of view of developing such systems is restructuring organizations in a process oriented way rather than function oriented [6]. Difference of process and function oriented enterprise structure is shown in Figure 5.

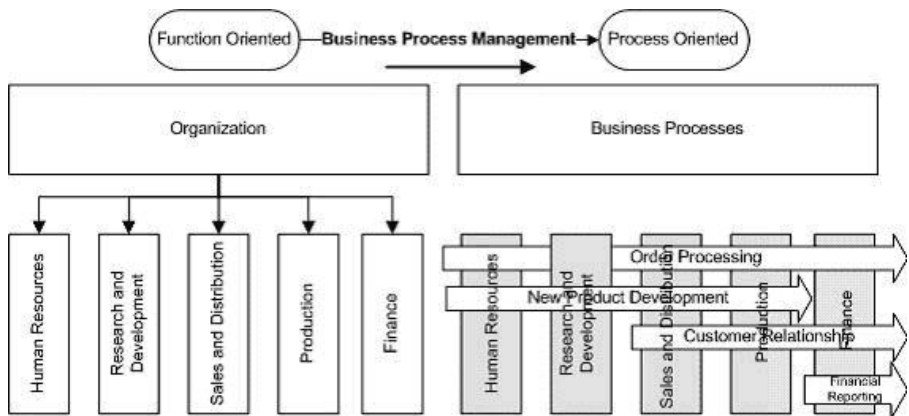


Fig. 5. Structures of process oriented and function oriented enterprises (Skok and Döringer 2001)

Today ERP systems are utilized as software applications. ERP systems can be defined in three different ways: [42] A commercial software product that is sold and purchased; (2) A management tool that held all of the data and processes of an organization; [42] A key factor that gives solutions to the infrastructure of business processes [43]. In this study, the focus will be on an ERP system as a software product; since, the primary goal is to evaluate quality of ERP software.

The term “enterprise” refers to every function of a system that supplies services and/or products. ERP software provides one database, one application, and one user interface for distributed functions of enterprises [44]. All properties of ERP systems are shown in Figure 6.

These distributed functions cover production planning, purchase, inventory control, finance, human resources and sales and marketing [45]. In the last decade, Customer Relationship Management (CRM), Supply Chain Management and web applications are added on ERP systems as advanced applications [35].

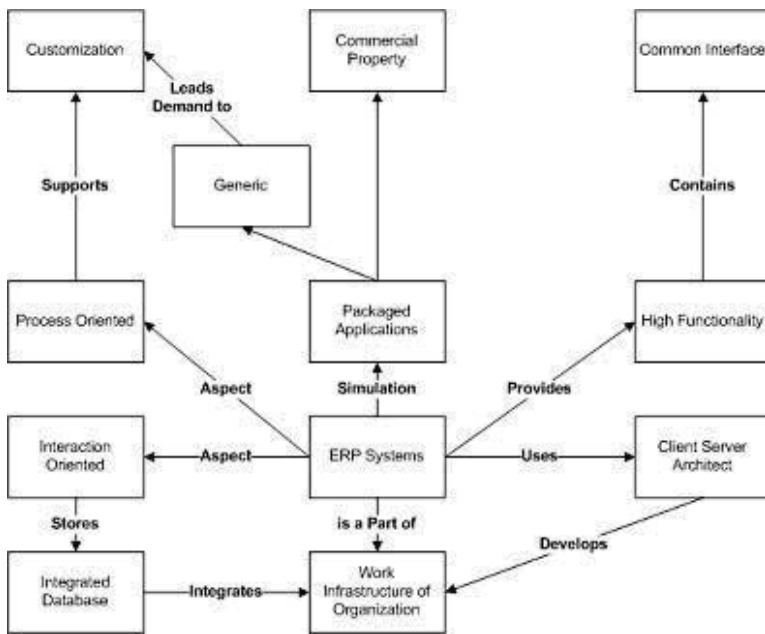


Fig. 6. Fundamental properties of ERP software [49]

Shields (2001) gives a framework for an eXtended Enterprise Systems (XES) and he defines XES as a systems that covers managerial portal, data warehouse, advanced applications, transactional backbone of ERP and a technical infrastructure. In reality ERP systems as software products also works on a technical infrastructure and those technical infrastructures include hardware, network, database management system, e-mail and gateway. On the other hand, it must be emphasized that in this study advanced applications and technical infrastructure are not going to be considered as a

part of ERP systems' core. In this study, we focus on transactional backbone of ERP systems; since, there are a lot of different advanced applications [35].

Definition of Shields [35] gives a good guidance for determining core of ERP systems. Following his guidance, it can be stated that the finance, sales and distribution, human resources and manufacturing components constitute transactional backbone of ERP [7, 35]. Visual representation of the transactional backbone of ERP systems is given in Figure 7.

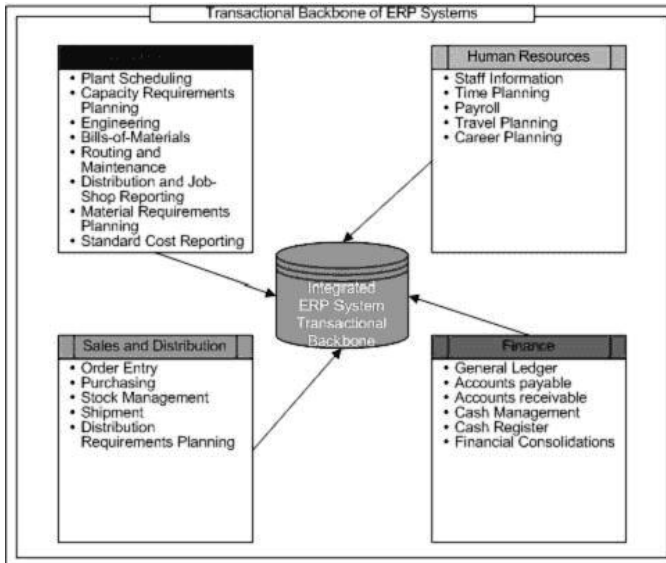


Fig. 7. Transactional backbone of ERP systems

It must be noted that ERP systems as a software product is not limited with the given components. Like the other software products, ERP systems also have security, authorization and help components and human-computer interfaces [43]. Moreover, these components are equally important for the users of software products. Therefore, while evaluating external quality of ERP, not only transactional backbone but also these additional components (security, authorization and help) must be considered.

4 Determining External Quality of ERP Software

It is obvious that the idea of ERP can satisfy the requirements of production systems. Although the idea of ERP systems is unique; ERP software products have many common features and also have significant differences. Therefore, end-users must consider related risk and evaluate these systems by considering conformance of particular software to their stated or implied needs.

Today, measuring External Quality Characteristics of a software product is considered as a valuable tool in Software Quality evaluation to determine conformance to the requirements of its users. In the beginning, External Software Quality

Characteristics can help them to examine software according to each Sub-Characteristic. In the frame of this reference, before buying any software product purchasers can test software and evaluate it according to external metrics.

When evaluating external metrics, end-users can grade software according to External Software Quality Sub-Characteristics. This grading can be a number between 0 and 100 as it is recommended in the standard. If this grade is denominated as E_{ij} , it can be explained as follows:

E_{ij} : Evaluation grade assigned by end-users to a Sub-Characteristic for software where

i : External Software Quality Characteristic ($i= 1, 2, 3, 4$)

j : External Software Quality Sub-Characteristic ($j= 1, \dots, J$)

On the other hand, while comparing different ERP software products it is hard to determine which ERP software best satisfies their requirements; because, each Quality Characteristic and Sub-Characteristic has different weights. With this respect, the present study tries to provide a guide by setting up a checklist of weighted Quality Characteristics to be used by organizations and/or users that intend to purchase ERP software. These external quality Sub-Characteristics are weighted according to requirement levels when considering core of ERP software. Afterwards, these weights and examination results of end-users can be used as a tool for determining External Quality of any ERP software.

To determine requirement levels, first a questionnaire is set up according to the formerly given External Software Quality Characteristics and Sub-Characteristics. Questions are set up in such manner that each one indicates a significant Software Quality Sub-Characteristic. Subsequently, five expert professors all of whose research areas of interest is Information Technology, Production Systems and ERP Systems are chosen to answer the questionnaire. While preparing questionnaire the following rules are taken as a guide:

1. The aim of the questionnaire is presented, a brief explanation is given and formerly defined transactional backbone of ERP software is elaborated
2. It is tried to be comprehensible while asking questions. Explanations are kept precise and brief.
3. It is tried to keep questionnaire short and attractive. It is tried not to reference to a previous question in any questions.
4. It is tried to structure the questionnaire well and experts are expected to follow its logical order.
5. Answers to the questions are pre-defined and each interviewee is expected to assign a value from 1 to 10 for each question which refers to a particular Sub-Characteristic. These values refer to a degree of requirement where 1 refers to lowest degree of requirement and 10 refer to highest degree of requirement. While assigning those values experts are expected to assign a high value to a question that is considered of high importance.
6. It is tried to keep questions in such a manner that they do not affect judgements of experts.
7. The experts are not allowed to impose their opinions by leading to specific answers while performing questionnaire.

Answers to these questions give weights of each external software quality sub-characteristics since each question indicates requirement levels of a significant External Software Quality Sub-Characteristic. The results are given in Table 3.

Table 3. Weights of each External Software Quality Sub-Characteristics

Charac-teristics (i)	Sub-Charac-teristics (j)	Ex-pert 1	Ex-pert 2	Ex-pert 3	Ex-pert 4	Ex-pert 5	Wij	Dij	Wi	Di
1. Func-tional-ity	1. Suit-ability	10	10	10	9	10	9,8	0,212	9,240	0,281
	2. Accu-racy	10	10	10	9	10	9,8	0,212		
	3. Inter-operabil-ity	8	10	10	9	9	9,2	0,199		
	4. Com-pliance	9	10	8	8	9	8,8	0,190		
	5. Secu-rity	9	10	8	8	8	8,6	0,186		
2. Reli-ability	1. Ma-turity	10	8	7	7	8	8,0	0,320	8,333	0,253
	2. Fault Toler-ance	8	10	9	9	7	8,6	0,344		
	3. Re-cover-ability	9	10	10	6	7	8,4	0,336		
3. Us-ability	1. Under-stand-ability	7	7	7	8	7	7,2	0,327	7,333	0,223
	2. Learn-ability	7	8	8	8	7	7,6	0,345		
	3. Oper-ability	6	8	8	8	6	7,2	0,327		
4. Effi-ciency	1. Time Behav-iour	8	8	9	8	9	8,4	0,525	8,000	0,243
	2. Re-source Behav-iour	7	8	8	8	7	7,6	0,475		

Notation for the formula below is explained subsequently:

W_{ij} : Weight of each Sub-Characteristic

D_{ij} : Relativistic degree of importance of each Sub-Characteristic of one Characteristic

W_i : Weight of each Characteristic

D_i : Relativistic degree of importance of each Characteristic among all Characteristics where

i : External Software Quality Characteristic ($i= 1, 2, 3, 4$)

j : External Software Quality Sub-Characteristic ($j= 1, \dots, J$)

Weight of each Sub-Characteristic [46] is the average of grades assigned by experts as seen in the formula. Therefore, weights of each Sub-Characteristic can be calculated by the following equation.

$$W_{ij} = \frac{\sum_{k=1}^K G_{kij}}{K}$$

where

G_{kij} : Grade assigned by the k th expert for the j th Sub-Characteristic of the i th Characteristics

k : Number of expert ($k= 1, \dots, 5$)

As it is seen in the table each Sub-Characteristics is related to one Characteristic relatively more important than one another. Therefore, to compare each Sub-Characteristics of one Characteristics relativistic degree of importance for each Sub-Characteristic can be calculated by using below formula.

$$D_{ij} = \frac{W_{ij}}{\sum_{j=1}^J W_{ij}}$$

where

D_{ij} : Relativistic degree of importance of a Sub-Characteristic for the i th Characteristic
By calculating average of Sub-Characteristics related to a Characteristic, weights of Characteristics can be found. For this purpose following equation can be utilized.

$$W_i = \frac{\sum_{j=1}^J W_{ij}}{J}$$

Moreover, relativistic degree of importance of each Characteristic can be calculated by using following formula.

$$D_i = \frac{W_i}{\sum_{i=1}^4 W_i}$$

where

W_i : Weight of the i th Characteristic.

D_i : Relativistic degree of importance of the i th Characteristic

As an example, the relativistic degree of importance of suitability Sub-Characteristic which is related to functionality is 21.2%. To give one more example it can be inferred that the efficiency quality characteristic has 24.3% effect on overall external quality when considering requirements of systems that intends to use ERP software.

This paper recommends ERP software buyers to use these weights before purchasing ERP software to find best alternative which meets their stated or implied needs. By using requirement levels, these weights help to evaluate External Software Quality of ERP software. For this purpose following equation can be used.

$$ESQoERP = \sum_i^4 \sum_j^J W_{ij} * E_{ij}$$

where

ESQoERP: Weighted External Software Quality of a particular ERP software

The given formula can be used to evaluate different ERP software products. If this approach is followed and alternatives are graded, selection can be made by choosing the software which has highest grade.

As it is stated before, transactional backbone of ERP software covers different components. Moreover, experts that contributed to this study recommended using one another variable; since, each module has different degree of importance dependent on the significant differences of the production systems. Especially, manufacturing and service systems have major differences. Therefore, in this study it is proposed to assign one more weight to each component for manufacturing and service systems. With this respect, in Table 4 weights of components for those systems are proposed.

Table 4. Proposed weights of components for manufacturing systems and service systems

Components (C)	Manufacturing Systems	Service Systems
1. Sales and Distribution Components	0.20	0.20
2. Manufacturing Components	0.30	0.25
3. Human Resources Components	0.15	0.20
4. Finance Components	0.25	0.25
5. Additional Components	0.10	0.10

If it is desired to weight each module individually by using these weights component based weighted ERP software product quality can be calculated by the subsequent formula.

$$CbESQoERP = \sum_l^5 C_l * \sum_i^4 \sum_j^J W_{ij} * E_{ij}$$

where

CbESQoERP: Component based weighted External Software Quality of particular ERP software

Cl: Weight of lth Component

l: Number of component (l= 1,...5)

In some cases due to special conditions according to the requirements of systems evaluators, different weights than proposed ones can be used. Under such circumstances, by following the below constraint evaluators can define their weights.

$$\sum_i^5 C_i = 1$$

Finally, by using the given formulas a particular ERP software product's overall external quality can be calculated. These evaluations will help the users to systematically evaluate ERP software and will give chance to benchmark different products.

5 Conclusion

In the field of IT, the most of the actions can be rolled back by its administrator or user. Only exception is implementation of ERP software within production systems. Purchasers of these products explore better ways to evaluate quality of these software products before making decision on buying one of them. With this respect, it is tried to determine the requirement levels of ERP software and weight them accordingly. By providing equations it is tried to provide a guide to make it possible to benchmark different ERP software products in a systematic manner. Even though proposed approach can be applied to any other software products, in this study ERP software is chosen; since, risk related to ERP software is more than other software.

In its current form, the present study proposes a model for evaluating external software quality of ERP software. In the near future, major ERP software can be evaluated in a number of production systems and compare the results to figure out the effectiveness of the model proposed. In addition to these, while using ERP software quality in use can become an important issue. Therefore, the main limitation of the current study is that it does not incorporate quality in use. In the near future, model for evaluating quality in use can be developed for ERP software.

References

1. Goldhard, J.: Business Strategies for the 21st Century Manufacturing Firm-Using CIM for Competitive Advantage. In: AUTOFACT, USA (1992)
2. Sohal, A.: A Longitudinal Study of Planning and Implementation of Advanced Manufacturing Technologies. *International Journal of Computer Integrated Manufacturing* 10(1-4), 81-95 (1997)
3. Nagalingam, S.V., Lin, G.C.I.: Latest Developments in CIM. *Robotics and Computer Integrated Manufacturing* 15, 423-430 (1999)
4. Da Silveria, G., Borenstein, D., Fogliatto, F.: Mass Customization: Literature Review and Research Directions. *International Journal of Production Economics* 72, 1-13 (2001)
5. Digre, T.: Business Application Components. In: *Object Oriented Programming Systems Languages Application*, Austin, USA (1995)
6. Skok, W., Döringer, H.: Potential Impact of Cultural Differences on Enterprise Resources Planning (ERP) Projects. *The Electronic Journal of Information Systems in Developing Countries* 7(5), 1-8 (2001)

7. Davenport, T.H.: Putting the Enterprise into the Enterprise System. *Harvard Business Review* 76(4), 121–132 (1998)
8. Kumar, V., Maheshwari, B., Kumar, U.: An Investigation of Critical Management Issues in ERP Implementation: Empirical Evidence from Canadian Organizations. *Technovation* (2001)
9. Markus, M.L., Tanis, C.: The Enterprise System Experience: From Adoption to Success. In: *Framing the Domains of IT Management: Projecting the Future through the Past*. Pinnaflex Educational Resources Inc., Cincinnati (2000)
10. Hong, K.K., Kim, Y.G.: The Critical Success Factors for ERP Implementation: An Organizational Fit Perspective. *Information & Management* (40), 25–40 (2002)
11. Wider, C., Davis, B.: False Starts, Strong Finishes. *Information week* 711, 41–53 (1998)
12. ISO, ISO 25000: Quality Software Requirements and Evaluation. The International Organization for Standardization, Geneva (2004)
13. ISO, ISO/IEC 9126: Product Quality - Part 1: Quality Model (2001)
14. Crosby, P.: *Quality is Free*. McGraw-Hill, New York (1979)
15. Calero, C.: *Handbook of Research on Web Information Systems Quality* (2008)
16. Olsina, L.: Measuring Web Application Quality with WebQEM. *IEEE Multimedia* (2002)
17. Feigenbaum, A.V.: *Total Quality Control*. McGraw-Hill, New York (1991)
18. Juran, J.: *Juran on Quality by Design*. The Free Press, New York (1992)
19. Xenos, M., Christodoulakis, D.: Evaluating Software Quality by the Use of User Satisfaction Measurements. In: *The 4th Software Quality Conference, University of Abertay Dundee* (1995)
20. Xenos, M.: Usability Perspective in Software Quality. In: *Usability Engineering Workshop, The 8th Panhellenic Conference on Informatics with International Participation, Southern Cyprus* (2001)
21. McCall, J.A., Richards, P.K., Walters, G.F.: *Factors in Software Quality*. Rome Air Development Centre, Rome (1977)
22. Boehm, B.W.: *Characteristics of Software Quality*. North Holland Publishing Co, New York (1978)
23. Bowen, T.P., Wigle, G.B., Tsai, J.T.: *Specification of Software Quality Attributes*. Rome Air Development Centre, Rome (1985)
24. Grady, R.B., Caswell, D.L.: *Software Metrics: Establishing a Company-Wide Program*. Prentice-Hall, London (1987)
25. Deutsch, M.S., Willis, R.R.: *Software Quality Engineering*. Prentice-Hall, London (1988)
26. Forse, T.: *Qualimétrie des Systèmes Complexes*. Les Editions d'Organisation (1989)
27. Von Maryhauser, A.: *Software Engineering Methods and Management*. Academic Press (1990)
28. Khoshgoftaar, T.M., Allen, E.B.: Classification Techniques for Predicting Software Quality: Lessons Learned. In: *Annual Oregon Workshop on Software Metrics, University of Idaho, USA* (1997)
29. ISO, ISO 9126: Product Quality - Part 2: External Metrics (2003)
30. ISO, ISO 9126: Product Quality - Part 3: Internal Metrics (2003)
31. ISO, ISO 9126: Product Quality - Part 4: Quality in Use Metrics (2004)
32. Dromey, R.G.: *Software Product Quality: Theory, Model and Practice*, Software Quality Institute, Brisbane, Australia (1998)
33. SEC, Knowledge Area: Software Quality Analysis, The Software Engineering Body of Knowledge (SWEBOK), Software Engineering Committee, Institute of Electrical and Electronics Engineers, Inc., Montreal (1999)

34. Stavrinoudis, D.: Early Estimation of Users' Perception of Software Quality. *Software Quality Journal* 13, 155–175 (2005)
35. Shields, M.G.: *E-Business and ERP - Rapid Implementation and Project Planning*. John Wiley & Sons, Inc., New York (2001)
36. Light, B., Holland, C.: *Enterprise Resource Planning Systems: Impacts and Future Directions*. In: *Systems Engineering for Business Process Change: Collected Papers from the EPSRC Research Programme*. Springer, London (2000)
37. Plenert, G.: Focusing Material Requirements Planning (MRP) towards Performance. *European Journal of Operational Research* 119, 91–99 (1999)
38. Browne, J., Harhen, J., Shirman, J.: *Production Management Systems*. Addison-Wesley (1988)
39. Hatzilygeroudis, I.: MRP II-Based Production Management Using Intelligent Decision Making. In: *Beyond Manufacturing Resource Planning. Advanced Models and Methods for Production Planning*. Springer (1998)
40. Greene, J.: *Production and Inventory Control Handbook*. McGraw-Hill (1987)
41. Hussein, J.: *Providing an Insight on Improving Performance of MRP*. Clemson University, Clemson (2000)
42. Hammer, M., Champy, J.: *Reengineering the Corporation: A Manifesto for Business Revolution*. Nicholas Brearley Publishing, London (1993)
43. Alageo, M.E.A., Barkmeyer, E.J.: *An Overview of Enterprise Resource Planning Systems in Manufacturing Enterprises*, National Institute of Standards and Technology (1999)
44. Cambashi, *Enterprise Resources Planning for Manufacturers*. Cambashi Ltd., Cambridge (1999)
45. Brislen, P., Krishnakumar, K.: *What is ERP, Enterprise Resource Planning* (1999)
46. Bertrand, J.W.M., Zuijderwijk, M., Hegge, H.M.H.: Using Hierarchical Psuedo Bills of Material for Customer Order Acceptance and Optimal Material Replenishment in Assemble to Order Manufacturing of Non-Modular Products. In: *International Journal of Production Economics* (2000)
47. Software Engineering Committee, Knowledge Area: Software Quality Analysis, The Software Engineering Body of Knowledge (SWEBOK). Institute of Electrical and Electronics Engineers, Inc., Montreal (1999)
48. Dromey, R.G.: *A Model for Software Product Quality*. IEEE Transactions on Software Engineering (1995)
49. Hagman, A.: *What will be of ERP? Could Component Software Spell a Strategic Inflection Point for the Industry?*, School of Information Systems Queensland University of Technology, Queensland (2000)

Towards a Catalog of Spreadsheet Smells^{*}

Jácome Cunha¹, João P. Fernandes^{1,2}, Hugo Ribeiro¹, and João Saraiva¹

¹ HASLab / INESC TEC, Universidade do Minho, Portugal
{jacome,jpaulo,jas}@di.uminho.pt, pg15970@alunos.uminho.pt
² Universidade do Porto, Portugal

Abstract. Spreadsheets are considered to be the most widely used programming language in the world, and reports have shown that 90% of real-world spreadsheets contain errors.

In this work, we try to identify spreadsheet smells, a concept adapted from software, which consists of a surface indication that usually corresponds to a deeper problem. Our smells have been integrated in a tool, and were computed for a large spreadsheet repository. Finally, the analysis of the results we obtained led to the refinement of our initial catalog.

Keywords: Spreadsheets, Code Smells, EUSES Corpus.

1 Introduction

Spreadsheets are widely used, specially by non-professional programmers, the also called "end users" [26]. A typical end user is teacher, an engineer, a student, or anyone that is not a professional programmer. The number of end-user programmers vastly outnumbers the amount of professional programmers. In fact, studies suggest that in the U.S. alone there exist 11 million end users against 2.75 million of professional programmers [30]. In the same study, it is projected for 2012 a total number of 90 million end users, 55 million of which will be working on spreadsheets or databases.

The number of spreadsheet users provides enough evidence that millions of spreadsheets are created every year. The fact is that, since end users are not professional programmers, they usually do not follow the principles of good programming. Instead, they care about getting a concrete task done. This approach, together with the lack of support for abstraction, testing, encapsulation, or structured programming in spreadsheets, leads to reports showing that up to 90% of real-world spreadsheets contain errors [29], with concrete impacts on companies' profits.

In this paper we present a catalog and a methodology to identify *smells* in spreadsheets. The concept of *code smell* (or simply bad smell) was introduced

^{*} This work is funded by the ERDF through the Programme COMPETE and by the Portuguese Government through FCT - Foundation for Science and Technology, project references [PTDC/EIA-CCO/108613/2008](#) and [PTDC/EIA-CCO/116796/2010](#). The two first authors were funded by FCT grants [SFRH/BPD/73358/2010](#) and [SFRH/BPD/46987/2008](#), respectively.

by Martin Fowler [18] as a concrete evidence that a piece of software may have a problem. Usually a smell is not an error in the program, but a characteristic that may cause problems understanding the software, for example, a long class in an object-oriented program, and updating and evolving the software. We present a methodology to define such smells in spreadsheets. We start by defining a set of possible smells as our initial catalog. Then we use a large repository to evaluate those smells. Finally, and as a result of this evaluation, we refine our spreadsheet smells in order to have a more robust catalog. To perform the detection of smells automatically we have developed a tool: SMELLSHEET DETECTIVE.

This paper is structured as follows. In Section 2 we present the methodology used to create, validate, evaluate and refine a catalog of bad smells for spreadsheets. Section 3 presents our initial catalog of bad smells. Then, in Section 5, we present the evaluation of our initial catalog in the EUSES corpus. In Section 4, we validate the results from the previous section. In Section 6 we adjust the initial catalog according to the results obtained. Section 7 introduces the developed tool to detect smells. In Section 8 we present related work and finally, Section 9 concludes the paper.

2 A Methodology to Identify Spreadsheet Smells

The detection of errors in software systems is an important software engineering technique. Software errors cause programs not to behave as expected and are responsible for several accidents. Even if not necessarily errors, the presence of bad smells in software code can make programs harder to understand, maintain, and evolve, for example. Martin Fowler popularized this notation of program smells in the context of object-oriented programming and this is now an important area of research. The detection of bad smells allows programmers to improve their programs by eliminating them.

In this section we present a methodology to define a catalog of bad smells for spreadsheets. This methodology is based on four steps: *catalog definition*, *catalog validation*, *catalog evaluation* and *catalog refinement*.

- Step 1: *Catalog Definition*** - based on our personal experiences, we propose an initial catalog of spreadsheet bad smells. We consider a bad smell in spreadsheets a reference in a formula to an empty cell, an empty cell in a table, etc. The full catalog is presented in Section 3.
- Step 2: *Catalog Validation*** - in order to validate the catalog we consider a large repository of spreadsheets, the EUSES corpus [21], that contains more than 5000 spreadsheets, and we detect smells in a representative sub-set of the repository, as described in Section 4.
- Step 3: *Catalog Evaluation*** - in order to evaluate the results of our empirical experiment performed in the previous step, we manually inspect all bad smells detected by our catalog. We classify the detected smells in four categories: *not a smell*, *low smell*, *medium smell* and *high smell*. We present this evaluation in detail in Section 5.

Step 4: *Catalog Refinement* - based on the evaluation performed in Step 3, we have adjusted our catalog by identifying wrong smells, by refining previously defined smells and by adding new smells that showed up when manually inspecting EUSES spreadsheets. The result of this step is our catalog of spreadsheet bad smells, which is shown in Section 6.

In order to validate our catalog in a large corpus we need a tool to automatically detect smells in spreadsheets. Thus, the full catalog defined in Step 1 was implemented as a software tool, SMELLSHEET DETECTIVE, that we present in Section 7.

3 Spreadsheet Smells: Catalog Definition

The notion of *bad smell* emerged from the need to identify the cases for which the internal structure of a piece of software could be improved. A bad smell is typically something that is easy to spot, and an indicator of a possible concrete issue. However, a smell is not something that can necessarily be considered an error. An example of a smell proposed by Fowler and that applies to a software project is the *long method smell*, that implements the notion that defining too long methods (i.e, methods larger than around a dozen lines of code) may lead to understandability and maintainability problems in the future.

The smells introduced by Fowler consist of a single flat list, but Mantyla has created a taxonomy for all those smells [25]. Similarly to what Mantyla has done we also grouped our smells in different categories: *Statistical Smells*, *Type Smells*, *Content Smells* and *Functional Dependencies Based Smells*.

In Figure 1 we present a spreadsheet, slightly adapted from a spreadsheet in the EUSES repository, where we can observe at least one smell belonging to each of the categories that we have defined. In the next sections, we present in detail the smells that we have fitted in each of these categories.

3.1 Statistical Smells

This category groups smells that are calculated through statistical analysis, namely the *Standard Deviation* smell.

– Standard Deviation

This smell detects, for a group of cells holding numerical values, the ones that do not follow their normal distribution.

Detection. Most spreadsheets with numeric values are organized either by rows or columns, and it is often the case that wrong values are introduced without the user ever noticing. The *Standard Deviation* smell is detected by analyzing the rows (or columns) of a spreadsheet and flagging the values outside the normal distribution of 95,4% (two standard deviations). In the detection of this smell neither formulas nor labels are taken into account.

	A	B	C	D	E	F	G	H	I	J
1	code	upc	description	size	store_nr	week	qty	price	onSale	profit
2	653	1111140009	DOVE DISH LIQUID	42 OZ	101	385	2	1.99		3.98
3	653	1111140009	DOVE DISH LIQUID	42 OZ	101	0	5	1.99		9.95
4	653	123	DOVE DISH LIQUID	42 OZ	101	387	4	1.99	=G * H	7.96
5	653	1111140009	DOVE DISH LIQUID	42 OZ	101	388	4	1.99		13.14
6	653	1111140009	DOVE DISH LIQUID	42 OZ	102	391	6	2.19		17.52
7	653	1111140009	DOVE DISH LIQUID	42 OZ	102	392	8	2.19		26.91
8	654	1111143002	SUNLIGHT DISH LIQUIDS	64 OZ	100	383	9	2.99		62.79
9	654	1111143002	SUNLIGHT DISH LIQUID	64 OZ	100	384	21	2.99		29.90
10	654	1111143002	SUNLIGHT DISH LIQUID	64 OZ	100	385	10	2.99		17.94
11	654	1111143002	SUNLIGHT DISH LIQUID	64 OZ	100	386	6	2.99		118.65
12	655	1111147006	SUNLIGHT POWDER AUTO	85 OZ	102	391	35	3.39	S	15.16
13	655	1111147006	SUNLIGHT POWDER AUTO	85 OZ	102	392	4	3.79		12.25
14	655	1111147006	SUNLIGHT POWDER AUTO	85 OZ	103	383	7	1.75		18.85
15	655	1111147006	SUNLIGHT POWDER AUTO	85 OZ	103	384	5	3.77		0.00
16	655	1111147006	SUNLIGHT POWDER AUTO	85 OZ	103	385		3.79		3.49
17	655	1111147006	SUNLIGHT POWDER AUTO	85 OZ	103	386	1	3.49		5.72
18	655	1111147006	SUNLIGHT POWDER AUTO	85 OZ	103	387	2	2.86		8.94
19	655	1111147006	SUNLIGHT POWDER AUTO	85 OZ	103	388	3	2.98		6.98
20	655	1111147006	SUNLIGHT POWDER AUTO	85 OZ	103	389	2	3.49		1.99
21	655	1111147006	SUNLIGHT POWDER AUTO	85 OZ	103	390	1	1.99		33.90
22	655	1111147006	SUNLIGHT POWDER AUTO	85 OZ	103	391	10	3.39	S	3.49
23	655	1111147006	SUNLIGHT POWDER AUTO	85 OZ	103	392	1	3.49		

Fig. 1. A spreadsheet for a warehouse of cleaning products

Example. By inspecting Figure 1 we have realized, for instance, that the standard deviation of column B values is 2.369E8. Then, the values that are acceptable by normal distribution should be within the range [5.868E8, 1.534E9]. This means that a smell is detected for cell B4, since it contains the value 123. Cell G12 is also indicated as a smell by standard deviation analysis.

3.2 Type Smells

In this category we have included the *Empty Cell* and the *Pattern Finder* smells, both analyzing the type of a cell, being it a *Label*, a *Number*, a *Formula* or an *Empty Cell*.

– Empty Cell

In order to detect cells that are left empty, but that occur in a context that suggests they should have been filled in, we have implemented the *Empty Cell* smell.

Detection. What we do here is to select all possible windows of five cells from each row (or column) and verify in each window whether it holds or not precisely one empty cell.

Example. In the spreadsheet in Figure 11, we can see that cells C6 and G16 are empty. However, they occur in a context where all their neighbor cells have been filled in; for these cells, a smell is signaled. Notice that, for example in column I, several other empty cells are not pointed out as smells: indeed, they occur in windows of 5 cells where more than just a single cell is empty.

– Pattern Finder

For faster development, spreadsheet users often simplify parts of formulas by introducing in them constant (numeric or label) values. This is a poor design decision that, if not corrected at some point, may lead to problems. In order to point out the cells where this situation occurs, we have implemented the *Pattern Finder* smell, that in fact works not only for formulas. Indeed, this smell is able of finding patterns in a sheet such as a row containing only numerical values except for one cell holding a label or a formula, or being empty. We flag such a cell as a smell.

Detection. The detection of *Pattern Finder* smell follows an approach in all similar to the detection of *Empty Cell* smells. In *Pattern Finder*, we use a four cell window and we search for one cell with type characteristics that are different from the ones in the other window cells. In this smell, we have chosen a smaller window since the occurrence of patterns other than empty cell patterns is also smaller.

Example. In Figure 11 we can see that cell F3 holds the textual value "o", being then a cell of type Label; furthermore, it is surrounded by numbers in the window constructed as described above: F3 is then a smell, and indeed it is likely that the value that instead should have been inserted is "0".

3.3 Content Smells

In this category we include smells that are found through the analysis of the content of cells: the *String Distance* smell and the *Reference to Blank Cells* smell.

– String Distance

Typographical errors are frequent when typing in a computer. In order to try to detect these type of errors, we have implemented the *String Distance* smell, that signals string cells that differ minimally with respect to other cells in a spreadsheet.

Detection. In the detection of this smell we use the algorithm created by Levenshtein [23]. This algorithm compares two strings and finds the minimum number of edits that are needed to transform one string into another.

In our case, we apply Levenshtein's algorithm to each pair of strings in a row (or column), and signal as a smell the cases for which the result is the value 1. This means that a single change to a string in the pair is enough to obtain the other string. We have furthermore limited the comparison to strings of length greater than three: this prevents, for example, all cells in a row holding the alphabet letters to be considered smells.

Example. We can see a *String Distance* smell in Figure 11 in row C, cell C8 holds the plural of cells C9 to C11; this suggests a typing error on cell C8.

– Reference to Empty Cells

The existence of formulas pointing to empty cells is a typical source of errors in spreadsheets, and we have therefore included in our catalog a smell for detecting all occurrences of this situation.

Detection. The detection of this smell is implemented by searching for all formulas in a spreadsheet and by gathering all their references. Then, we simply check whether each of these references points to an empty cell or not.

Example. Cell J16 of Figure 11 is recognized as a smell since its value is calculated using the value of cell G16, which is empty.

3.4 Functional Dependencies Based Smells

In this category, we have adapted to spreadsheets data mining techniques that were first introduced for databases. In particular, we search for *dirty values* in a spreadsheet.

– Quasi-Functional Dependencies (QFD)

In [4] it is described a technique to identify dirty values using a slightly relaxed version of *Functional Dependencies* [5], and this is the technique that we use here. Two columns A and B are *Functionally Dependent* if multiple occurrences of the same value in A always correspond to the same value in B. For instance, in the example show in Figure 11, column A functionally determines column B, as knowing one particular **code** is enough to know its associated **upc**. When equal values in a column correspond to the same value in another column, except for a *small* number of cases, this is a situation that *smells*, and that we flag.

Detection. The implementation of this smell follows the approach described in [4]. It involves collecting and matching all data in a spreadsheet in order to find QFD, actually identifying dirty values and then ranking all these values.

Example. Cells E12 and E13 of the spreadsheet presented in Figure 1 are pointed out as smells by the analysis of QFD. Indeed, we may see that the values (655, 1111147006, *SUNLIGHT POWDER AUTO*, 85 OZ) in columns A to F always determine the value 103 in column E, except precisely for cells E12 and E13.

4 Catalog Validation

In the previous section, we have described the catalog of spreadsheet smells that we propose in this paper. Now, we need to validate our catalog against real spreadsheets. For this purpose, we will use the EUSES Corpus [21], a large spreadsheet repository which has been widely used by the software engineering community [11,7]. This repository consists of 5606 spreadsheets which have been divided into six categories: Database, Financial, Grades, Homework, Inventory and Modeling. The proportion of spreadsheets per category can be seen in Figure 2.

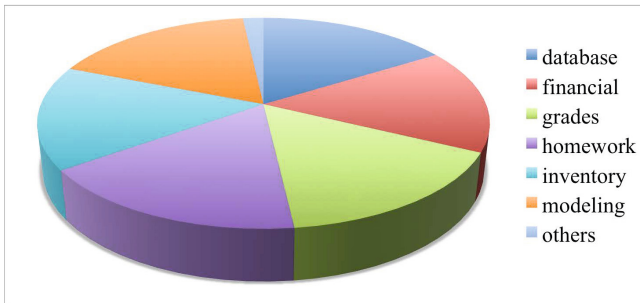


Fig. 2. EUSES spreadsheet categories

In order to evaluate our catalog, we have implemented a tool, SMELL-SHEET DETECTIVE, the spreadsheet smells that the catalog includes (this tool is presented in detail in Section 7). For each smell, the results we obtain are presented in Table 1: the different smells are listed in rows and the different EUSES categories are listed in columns.

The results observed in Table 1 consider 180 EUSES spreadsheets *only*. Still, smells were identified, in those spreadsheets, 3841 times. This is a number that is large enough for validation purposes while still enabling the manual validation of the detected smells. Indeed, the next phase of our methodology was to inspect individually and in detail each spreadsheet that we considered, and validate the smells identified for them. The validation tried to establish two things: a) whether the identification of a smell was accurate or not; b) in case the identified

Table 1. Number of smells collected for each EUSES category

Smell/Category	Database	Financial	Grades	Homework	Inventory	Modeling	Total
Empty cells	53	27	56	42	109	106	393
Patterns	86	62	66	58	111	114	497
Std. Dev.	33	35	134	14	32	7	255
String Dist.	25	3	1357	231	158	658	2442
CFD	12	13	150	2	13	24	204
Ref2empty	0	23	27	0	0	0	50
Total:	209	163	1790	347	423	909	3841

smell was accurately pointed out, how *severe* we consider it to be, i.e., how much a smell impacts in the overall quality of the spreadsheet.

5 Catalog Evaluation

The purpose of this section is to establish a model to evaluate the smells that our catalog identifies as such. For this, we need to define how the smells that are identified can be measured and classified. We have inspired ourselves in the technique used by the software quality measurement company Software Improvement Group [19] and empirically classify the smells that we detect under three categories:

- **Low smells (ls):** In this category we include the cells that are identified as smells but that we can not ensure that indeed constitute smells, neither can we ensure that, on the contrary, do not constitute smells;
- **Medium smells (ms):** This category includes the cells that we can ensure are smells with a high degree of certainty, but that belong to spreadsheets that we can not fully understand;
- **High smells (hs):** Here, we include all smells that are detected and that our inspection confirmed as such, without a doubt;
- **Not smells (ns):** Finally, we include here all smells that we consider to have been wrongly detected, for example, due to tool malfunctioning;

The creation of this classification model made it possible to uniformly classify spreadsheet smells. In the remaining of this section, we present different views of the classification we establish for each smell we identify.

In Table 2, we show the absolute results of classifying each detected smell under one category of the catalog validation model. As a concrete example of this, the value 7 in line 'Empty Cells', column 'Database', sub-column 'ls' indicates that 7 of the smells that were identified by the 'Empty cells' smell in spreadsheets of the 'Databases' EUSES category were identified as low smells.

Table 2. Smell classification (absolute values)

Smell/Level	Database				Financial				Grades				Homework				Inventory				Modeling			
	ls	ms	hs	ns	ls	ms	hs	ns	ls	ms	hs	ns	ls	ms	hs	ns	ls	ms	hs	ns	ls	ms	hs	ns
Empty cells	7	1	0	45	3	2	0	22	15	1	0	40	1	0	0	41	1	0	0	108	88	0	0	18
Patterns	7	1	0	78	6	2	0	54	16	5	0	45	1	1	0	56	1	0	0	110	93	0	0	21
Std. Dev.	10	0	0	23	2	0	0	33	2	0	0	132	5	0	0	9	1	0	0	31	1	0	0	6
String Dist.	0	0	1	24	0	0	0	13	0	0	4	1353	1	0	0	230	1	0	2	155	11	2	0	645
CFDs	4	7	1	0	0	0	3	0	55	5	1	89	1	0	0	1	1	1	0	11	3	0	1	20
Ref2empty	0	0	0	0	9	1	7	6	0	8	1	18	0	0	0	0	0	0	0	0	0	0	0	0
Total:	28	9	2	170	20	5	10	128	88	19	6	1677	9	1	0	337	5	1	2	415	196	2	1	710

Table 3. Smell classification (relative values, per smell)

Smell/Level	Database				Financial				Grades				Homework				Inventory				Modeling			
	ls	ms	hs	ns	ls	ms	hs	ns	ls	ms	hs	ns	ls	ms	hs	ns	ls	ms	hs	ns	ls	ms	hs	ns
Empty cells	13	2	0	85	11	7	0	82	27	2	0	71	2	0	0	98	1	0	0	99	83	0	0	17
Patterns	8	1	0	91	10	3	0	87	24	8	0	68	2	2	0	96	1	0	0	99	82	0	0	18
Std. Dev.	30	0	0	70	6	0	0	94	1	0	0	99	36	0	0	64	3	0	0	97	14	0	0	86
String Dist.	0	0	4	96	0	0	0	100	0	0	0	100	0	0	0	100	1	0	1	98	2	0	0	98
QFD	33	59	8	0	0	0	100	0	37	3	1	59	50	0	50	8	8	0	84	13	0	4	83	
Ref2empty	0	0	0	0	40	4	30	26	0	30	4	66	0	0	0	0	0	0	0	0	0	0	0	0
Total:	13	4	1	81	12	3	6	79	5	1	0	94	3	0	0	97	1	0	0	98	22	0	0	78

Table 4. Smell classification (relative values, per evaluation model category)

Smell/Level	Database				Financial				Grades				Homework				Inventory				Modeling			
	ls	ms	hs	ns	ls	ms	hs	ns	ls	ms	hs	ns	ls	ms	hs	ns	ls	ms	hs	ns	ls	ms	hs	ns
Empty cells	25	11	0	26	15	40	0	17	17	5	0	2	11	0	0	12	20	0	0	26	45	0	0	3
Patterns	25	11	0	46	30	40	0	42	18	26	0	3	11	100	0	17	20	0	0	27	46	0	0	3
Std. Dev.	36	0	0	14	10	0	0	26	2	0	0	8	56	0	0	3	20	0	0	7	1	0	0	1
String Dist.	0	0	50	14	0	0	0	10	0	0	66	81	11	0	0	68	20	0	100	37	6	100	0	90
QFD	14	78	50	0	0	0	30	0	63	26	17	5	11	0	0	0	20	100	0	3	2	0	100	3
Ref2empty	0	0	0	0	45	20	70	5	0	43	17	1	0	0	0	0	0	0	0	0	0	0	0	0

In Table 3 we present the percentage of automatic detections, per smell, that were classified in each of the validation model categories. Taking the same cell as above, it means that 13% of empty cell smells that were identified in database spreadsheets were classified as low smells.

The information of Table 4 shows the percentage of automatic detections, per validation model category, for all the smells. Again reading the same table element, 25% of the smells that we have included in the 'Low Smells' category, for database spreadsheets, were identified by the 'Empty Cell' smell.

Finally, in Table 5, we present the total number of smells, for each of smell and for each model evaluation category.

Table 5. Global total results (absolute values)

Smell \ Level	Low smells	Medium smells	High smells	Not smells	Total
Empty cells	115	4	0	274	393
Patterns	124	9	0	364	497
Std. Dev. cells	21	0	0	234	255
String Dist.	13	2	7	2420	2442
QFD	64	13	6	121	204
Ref2empty	9	9	8	24	50
Total:	346	37	21	3437	3841

This table shows that we are able to detect smells in the EUSES corpus. However, 90% of the smelly cells detected were not confirmed as such after the manual inspection of those 3841 cells. In the next section we conduct a refinement on the initial catalog that we have defined, based on the analysis of the results presented in this section.

6 Catalog Refinement

Having evaluated our initial spreadsheet catalog in a representative set of spreadsheets of the EUSES repository, firstly, by automatically running a spreadsheet smell detector and, secondly, by manually validating the results, we can now refine the catalog based on the results and experience obtained. We consider three different types of refinement:

- *Overlapped smells:* The pattern smell overlaps the empty cell smell. In fact, all empty cells were detected by both approaches. Since we want to distinguish an empty cell from a pattern in order to have a more precise notion of the smell itself, then the empty cells should not be considered as a pattern smell.
- *New smells detected:* when manually inspecting a large number of EUSES spreadsheets, we detected several bad smells in spreadsheets that we had not considered, namely, the use of summation cells where no formulas are used to do the calculations: constant values are used instead! This is the case, for example, in the spreadsheet file “*FIN_hospitaldataset2002 MEMORIAL*”, cell F22. Thus, a smell detecting constant values where formulas should be used needs to be added to the catalog.
- *Wrong smells:* The string distances smell produced the highest number of wrongly detected cells. Because we use the Levenshtein distance algorithm to find close strings, most of the wrong string distance detected came from the use of the algorithm in numeric strings or strings with numeric values. One solution for this problem would be the identification of the character

where the strings were different and if that character was a numeric value we would ignore it. This would lead to an improvement in this smell of 88%. The other smells where the results would be improved is the standard deviation smell: in this one we do not have a percentage of improvement because most of the wrong detections was due the lack of knowledge of the domain of the spreadsheet.

Next we present the results of running our smell detector according to the catalog refinements. It should be noticed that no new smells have been implemented.

Table 6. Refined catalog results

Smell \ Level	Low smells	Medium smells	High smells	Not smells	Total
Empty cells	115	4	0	274	393
Patterns	9	5	0	90	101
Std. Dev. cells	21	0	0	234	255
String Dist.	13	2	7	290	312
QFD	64	13	6	121	204
Ref2empty	9	9	8	24	50
Total:	231	33	21	1033	1315

As the results included in Table 6 show, we were able to detect 285 manually validated smells out of 1315 detected by our smell detector. Thus we have 21.7% of correct smells detected. Moreover, 54 out of 231 (18.9%) presents a medium or high level probability of the occurrence of a problem in the spreadsheet. As a result of our refinement, we have improved our smell detection from 90% to 78% of false positive smells.

Based on our analysis we present in Table 7 the EUSES categories for which we believe each smell can be effectively used.

Table 7. The use of smells in EUSES categories

Smell \ Category	Database	Financial	Grades	Homework	Inventory	Modeling
Empty cells						X
Patterns						X
Std. Dev.		X				
String Dist.	X	X	X	X	X	X
QFD	X	X	X			X
Ref2empty	X	X	X	X	X	X

The table show that all smells are applicable in at least two categories and two of them (string distance and references to empty cells) are applicable in all categories.

7 SMELLSHEET DETECTIVE

To analyze the selected spreadsheet sample we implemented the tool **SMELLSHEET DETECTIVE** which detects the smells introduced in the Section 3. This implementation was made using the Java programming language, the Google Web Toolkit (GWT)¹, the Apache POI library² and the Google libraries to work with spreadsheets from the Google Docs³. We decided to support spreadsheets written in the Google Docs platform because it is becoming more and more used. In fact, the popular Microsoft Office suite has also its online version⁴. Indeed, the migration from desktop to online applications is becoming very common. Nevertheless, we also support spreadsheet written using desktop applications, as can be seen in Figure 3.

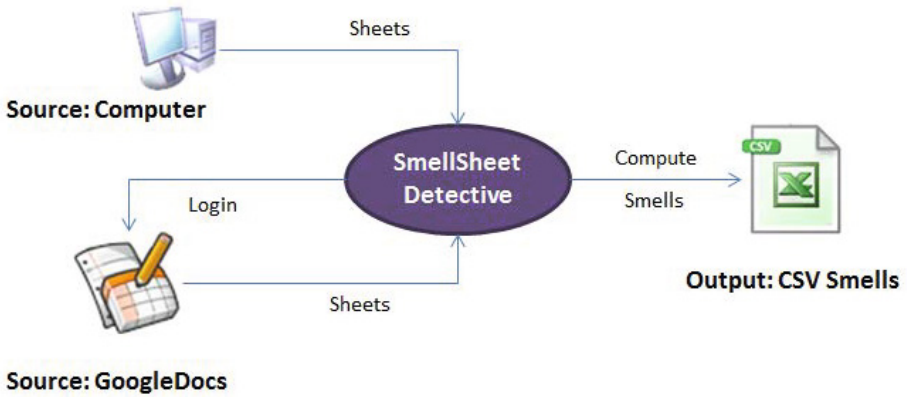


Fig. 3. SMELLSHEET DETECTIVE architecture

The tool can receive spreadsheets, as we mentioned before, from Google Docs or directly from the computer where it is running. If the Google Docs source is used, a valid Google account login is required. The tool then selects all the spreadsheets in the account and shows them to the user. After selecting a particular spreadsheet, the user can select a single sheet to be analyzed, or otherwise the entire spreadsheet is used. If we use the direct upload source, the user can browse the spreadsheets in the computer and select one. The sheet selection works in a similar way as explained.

After the selection process, the **SMELLSHEET DETECTIVE** searches the sheets selected for bad smells. The detection of smells is done as explained in Section 3.

Finally, the tool generates a comma-separated value file with the outputs. Each row will contain information about a single sheet: the first value is the

¹ <http://code.google.com/intl/pt-PT/webtoolkit/>

² <http://poi.apache.org>

³ <http://docs.google.com>

⁴ <http://www.officelive.com>

name of the spreadsheet/sheet and the next numbers are the ones reported by the different parts of the tool implementing the detection of the smells empty cells, standard deviation, string distance, functional dependencies, pattern finder and references to blank cells.

8 Related Work

Fowler [18] was the first to introduce the concept of smell, to create a list of 22 smells and pointing a possible solution for each one of them. In the sequence of Fowler study, Mantyla *et al.* [24] has created a taxonomy for the smells listed by Fowler so they could be easier to understand. They created five groups of smells, namely, the bloaters, the object-oriented abusers, the change preventers, the dispensables and the couplers.

In recent years is becoming a very active area of research. Engels and Erwig introduced *ClassSheets* to model the business logic of a spreadsheet [15]. These models have been extended and embedded in spreadsheet system [10,11] so that they can guide end users introducing correct data [12] and to provide model-driven software evolution [8,9,14]. Tools have been defined to debug spreadsheets [2,3,22], to define type systems for spreadsheets [16], to map spreadsheets to databases [13] and to infer models from legacy spreadsheets [7,20].

Hermans *et al.* [17] adapted Fowler's work to detect inter-worksheets smells. Their work differs from the work we present here in the fundamental approach to define spreadsheet smells: while Hermans adapts Fowler's smells to the spreadsheet realm, we analyze a large corpus, and based on that, we define, validate, evaluate and refine spreadsheet specific smells. We believe that the two approaches are orthogonal and as a consequence a full catalog should include both smells.

The analysis of possible errors in spreadsheets was also studied by several researchers, namely, Panko *et al.* [27] who proposed a revised taxonomy for spreadsheet errors [28], Correia *et al.* [6] who used Goal Question Metric to measure the maintainability of spreadsheets, and Abraham *et al.* [2] who developed a tool for debugging spreadsheets.

9 Conclusion

In this paper we have presented a detailed study on spreadsheet smell detection. We have presented a catalog of smells that are spreadsheet specific and we have validated it using a large spreadsheet repository. Furthermore, we have refined the catalog by manually evaluating the results obtained. The smell detection has been implemented in the SMELLSHEET DETECTIVE tool, and we have presented our preliminary results that show that we are able to detect a significant amount of smells using our tool. Also, we have confirmed that more than 20% of the detected smelly cells point to a possible problem in the spreadsheet.

In the future, we intend to extend the work presented in this paper in two different ways. Firstly, we believe that the number and the type of smells that

belong to a spreadsheet smell catalog can be further extended. Secondly, ever since the smells were first proposed for software repositories, they are usually associated with refactorings that can eliminate them. These are promising research directions that we are already exploring and whose results we plan to bring out in the near future.

References

1. Abraham, R., Erwig, M.: Inferring templates from spreadsheets. In: Proc. of the 28th Int. Conf. on Software Engineering, pp. 182–191. ACM, New York (2006)
2. Abraham, R., Erwig, M.: Goaldebug: A spreadsheet debugger for end users. In: ICSE 2007: Proceedings of the 29th International Conference on Software Engineering, pp. 251–260. IEEE Computer Society, Washington, DC (2007)
3. Abreu, R., Riboira, A., Wotawa, F.: Constraint-based debugging of spreadsheets. In: Proceedings of the XV Ibero-American Conference on Software Engineering (CibSE 2012) (to appear, 2012)
4. Chiang, F., Miller, R.J.: Discovering data quality rules. The Proceedings of the VLDB Endowment 1, 1166–1177 (2008)
5. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM 13(6), 377–387 (1970)
6. Correia, J.P., Ferreira, M.A.: Measuring maintainability of spreadsheets in the wild. In: ICSM, pp. 516–519. IEEE (2011)
7. Cunha, J., Erwig, M., Saraiva, J.: Automatically inferring classsheet models from spreadsheets. In: 2010 IEEE Symposium on Visual Languages and Human-Centric Computing, pp. 93–100. IEEE Computer Society (2010)
8. Cunha, J., Fernandes, J.P., Mendes, J., Pacheco, H., Saraiva, J.: Bidirectional Transformation of Model-Driven Spreadsheets. In: Hu, Z., de Lara, J. (eds.) ICMT 2012. LNCS, vol. 7307, pp. 105–120. Springer, Heidelberg (2012)
9. Cunha, J., Fernandes, J.P., Mendes, J., Saraiva, J.: MDSheet: A Framework for Model-driven Spreadsheet Engineering. In: Proceedings of the 34rd International Conference on Software Engineering, ICSE 2012, pp. 1412–1415. ACM (2012)
10. Cunha, J., Fernandes, J.P., Saraiva, J.: From Relational ClassSheets to UML+OCL. In: The Software Engineering Track at the 27th Annual ACM Symposium On Applied Computing (SAC 2012), Riva del Garda (Trento), Italy, pp. 1151–1158 (2012)
11. Cunha, J., Mendes, J., Fernandes, J.P., Saraiva, J.: Embedding and evolution of spreadsheet models in spreadsheet systems. In: IEEE Symp. on Visual Languages and Human-Centric Computing, pp. 179–186. IEEE CS (2011)
12. Cunha, J., Saraiva, J., Visser, J.: Discovery-based edit assistance for spreadsheets. In: Proceedings of the 2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), VLHCC 2009, pp. 233–237. IEEE Computer Society, Washington, DC (2009)
13. Cunha, J., Saraiva, J., Visser, J.: From spreadsheets to relational databases and back. In: Proceedings of the 2009 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, PEPM 2009, pp. 179–188. ACM, New York (2009)
14. Cunha, J., Visser, J., Alves, T., Saraiva, J.: Type-Safe Evolution of Spreadsheets. In: Giannakopoulou, D., Orejas, F. (eds.) FASE 2011. LNCS, vol. 6603, pp. 186–201. Springer, Heidelberg (2011)

15. Engels, G., Erwig, M.: ClassSheets: automatic generation of spreadsheet applications from object-oriented specifications. In: 20th IEEE/ACM Int. Conf. on Automated Sof. Eng., Long Beach, USA, pp. 124–133. ACM (2005)
16. Erwig, M., Burnett, M.: Adding Apples and Oranges. In: Adsul, B., Ramakrishnan, C.R. (eds.) PADL 2002. LNCS, vol. 2257, pp. 173–191. Springer, Heidelberg (2002)
17. Felienne Hermans, M.P., van Deursen, A.: Detecting and visualizing interworksheets smells in spreadsheets. In: Proceedings of the 34rd International Conference on Software Engineering, ICSE 2012. ACM (to appear, 2012)
18. Fowler, M.: Refactoring: Improving the Design of Existing Code. Addison-Wesley, Boston (1999)
19. Heitlager, I., Kuipers, T., Visser, J.: A practical model for measuring maintainability. In: Proceedings of the 6th International Conference on Quality of Information and Communications Technology, pp. 30–39. IEEE Computer Society, Washington, DC (2007)
20. Hermans, F., Pinzger, M., van Deursen, A.: Automatically Extracting Class Diagrams from Spreadsheets. In: D’Hondt, T. (ed.) ECOOP 2010. LNCS, vol. 6183, pp. 52–75. Springer, Heidelberg (2010)
21. Ii, M.F., Rothermel, G.: The euses spreadsheet corpus: A shared resource for supporting experimentation with spreadsheet dependability mechanisms. In: 1st Workshop on End-User Software Engineering, St. Louis, Missouri, USA, pp. 47–51 (2005)
22. Janssen, T., Abreu, R., van Gemund, A.: Zoltar: A toolset for automatic fault localization. In: 24th IEEE/ACM International Conference on Automated Software Engineering, ASE 2009, pp. 662–664 (November 2009)
23. Levenshtein, V.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady 10, 707 (1966)
24. Mäntylä, M., Vanhanen, J., Lassenius, C.: A taxonomy and an initial empirical study of bad smells in code. In: Proceedings of the International Conference on Software Maintenance, ICSM 2003, pp. 381–384. IEEE Computer Society, Washington, DC (2003)
25. Mäntylä, M.V., Lassenius, C.: Subjective evaluation of software evolvability using code smells: An empirical study. Empirical Softw. Engg. 11, 395–431 (2006)
26. Nardi, B.A.: A Small Matter of Programming: Perspectives on End User Computing, 1st edn. MIT Press, Cambridge (1993)
27. Panko, R.R., Aurigemma, S.: Revising the panko-halverson taxonomy of spreadsheet errors. Decision Support System 49, 235–244 (2010)
28. Panko, R.R., Halverson Jr., R.P.: Spreadsheets on trial: A survey of research on spreadsheet risks. In: Proceedings of the 29th Hawaii International Conference on System Sciences, HICSS 1996. Decision Support and Knowledge-Based Systems, vol. 2, pp. 326–335. IEEE Computer Society, Washington, DC (1996)
29. Rajalingham, K., Chadwick, D.R., Knight, B.: Classification of spreadsheet errors. In: Symposium of the European Spreadsheet Risks Interest Group (EuSprIG), Amsterdam (2001)
30. Scaffidi, C., Shaw, M., Myers, B.: The ‘55m end-user programmers’ estimate revisited. Tech. rep., Carnegie Mellon University, Pittsburgh (2005)

Program and Aspect Metrics for MATLAB*

Pedro Martins¹, Paulo Lopes¹, João P. Fernandes^{1,2},
João Saraiva¹, and João M. P. Cardoso²

¹ HASLab / INESC TEC, Universidade do Minho, Portugal
{`prmartins`,`plopes`,`jpaulo`,`jas`}@di.uminho.pt

² Universidade do Porto, Faculdade de Engenharia, Departamento de Eng.
Informatica, Porto, Portugal
`jmpc@fe.up.pt`

Abstract. In this paper we present the main concepts of a domain-specific aspect language for specifying cross-cutting concerns of MATLAB programs, together with a suite of metrics that is capable of assessing the overall advantage of introducing aspects in the development cycle of MATLAB software. We present the results of using our own suite to quantify the advantages of using aspect oriented programming, both in terms of programming effort and code quality. The results are promising and show a good potential for aspect oriented programming in MATLAB while our suite proves to be capable of analyzing the overall characteristics of MATLAB solutions and providing interesting results about them.

Keywords: Aspect Oriented Programming, Matlab, Aspect Metrics.

1 Introduction

MATLAB [1] is a high-level, interpreted, mathematically-oriented domain-specific language which has some key characteristics such as being based on matrix data types, not requiring variables declaration and including operator overloading. Combined with function polymorphism and dynamic type specialization these features, together with the MATLAB environment provided by *MathWorks* [2], create an interesting environment to easily model and simulate complex systems.

The fact is that, in MATLAB as in most programming languages, tasks such as exploiting non-uniform fixed-point representations, monitoring certain variables or including handlers to observe specific behaviors are extremely cumbersome, error-prone and tedious tasks. Indeed, each time one of these features is necessary, invasive changes to the original MATLAB program need to be performed.

* This work is funded by the ERDF through the Programme COMPETE and by the Portuguese Government through FCT - Foundation for Science and Technology, projects ref. PTDC/EIA/70271/2006 and PTDC/EIA-CCO/108995/2008. The first and third authors were also supported by FCT grants B13-2011/PTDC/EIA/70271/2006 and SFRH/BPD/46987/2008, respectively.

In addition, as MATLAB allows a higher-level of abstraction than, for example, the C programming language, the *de facto* standard embedded systems programming language, aspect rules may also be used to specialize the MATLAB input program to different target architectures. These specializations may include data types, array shapes, and implementations of a given function, and also contribute to the deterioration of the overall quality of MATLAB code.

The goal of this paper is three fold: firstly, we provide software complexity metrics for MATLAB programs. These metrics, based on the Halsted's complexity metrics [3], provide a quantification of the overall quality of a MATLAB program. Secondly, we present a set of metrics for an aspect oriented extension of MATLAB, and thirdly, we also present aspect metrics to quantify the quality of aspect MATLAB programs. Finally, we present our experimental results of using both suites of metrics on real MATLAB code and aspect MATLAB programs. We show the results we obtained by applying our metrics and then we analyze the impact of using aspects. We also show, using concrete examples, how the quality of a MATLAB program can see significant increases when an aspects language is used.

In [4], the authors have suggested an aspects language, and a compiler for it, that is capable of concern modularization and supports specific scientific computation tasks. Being aware of this Aspects Oriented Programming (AOP) approach to MATLAB, we use our own aspects language because we are familiar with it, because it is simple to use and also because it supports the creation of aspect-oriented versions of MATLAB programs. In this paper we argue that MATLAB programming can suffer from concerns pollution and that introducing concerns separation makes programming easier, faster and can help producing code with more quality. Nevertheless, those improvements are orthogonal to any aspect oriented approach.

The remainder of this paper is organized as follows. In Section 2 we introduce the MATLAB programming environment and two practical examples. In Section 3 we introduce our aspect oriented approach, which we use to show how the examples of the previous section can be improved. In Section 4 we introduce our suite of metrics to assess the quality of both MATLAB programs and their aspectualized versions. In Section 5 we show the results of applying the metrics to a set of MATLAB programs, and in Section 7 we conclude this paper.

2 MATLAB

MATLAB is a high-level matrix-oriented programming language to implement computationally intensive tasks faster than with traditional programming languages. The code presented below is an excerpt of a MATLAB function that implements the Discrete Fourier Transform (function `dft_custom`).

Example of a MATLAB Program:

```
function [mag] = dft_custom(x, n)
twoPI = 2.0 * pi;
n2 = n/2;
for i = 1:n2
```

```

    xre(i) = x(i);
end
for i = 0:n-1
    arg = twoPI * i / n;
...

```

There are several characteristics to notice regarding MATLAB code. Firstly, to provide faster development, data types and shapes of variables are not specified. In the particular case of this example, from the assignment `twoPI = 2 * pi`, one could assume `twoPI` would be stored as a scalar of double-precision floating-point type¹. Internally, `twoPI` will actually be stored as a single-element array of type double.

Secondly, array variable shapes are inferred during program execution. Indeed, the assignments to variables expose, at runtime, the shape of those variables. At a certain point of a MATLAB program, assigning `arg = twoPI * i / n` will imply that `arg` is a single-element array, while at another point of the code assigning `arg=[1 2; 3 4; 5 6]` will imply that `arg` now refers to a 2×2 matrix. While these dynamic features speed up program development, they complicate the translation of MATLAB to non-dynamic languages. For many systems, the overhead to implement this dynamic behavior is just not acceptable.

In addition to functions, MATLAB offers the possibility of structuring code in scripts, which very much resemble bash scripts in UNIX (a simple sequence of instructions), or even Object Oriented Programming (OOP), by defining classes and applying standard object-oriented design patterns which allow code reuse, inheritance, encapsulation, and reference behavior. Despite promising, the introduction of OOP in MATLAB is still very recent and has not yet seen a broad use, whereas functions are part of most existing MATLAB solutions.

Despite some syntactic similarities with C, semantically they are very different. For example, the variable whose value is returned by a function, `mag` in the case of the previous example, is declared in the function signature itself, instead of by a primitive such as `return`, as in C; also, MATLAB functions may explicitly return more than one value.

Finally, function calling in MATLAB is easy in the sense that all functions are polymorphic, so a programmer knows that whatever variable he/she applies to a function, it will always produce a concrete result. Take as an example the `×` operator, that can be used to multiply integers, floats and even matrices.

2.1 Tracing in MATLAB

Tracing is a specialized method to obtain execution information of a program, and it is a technique that is frequently used during the debugging of a program.

A concrete scenario where tracing would be of practical interest is the following: in case a function, such as `dft_custom`, is not producing the expected results, the programmer may want to print all values assigned to some variables when

¹ `pi` is a constant in MATLAB representing the value π .

executing that function for a particular input. Observing all the printed values may then help in understanding and identifying the coding error.

In order to implement this tracing, one could follow the strategy of planting printing instructions throughout the original source code, every time an assignment occurs. For `dft_custom`, this strategy would result in the code shown next:

Example of a MATLAB Program with tracing:

```
function [mag] = dft_custom_tracing(x, n)
twoPI = 2.0 * pi;
disp('The value of twoPI is:'); disp(twoPI);
n2 = n/2;
disp('The value of n2 is:'); disp(n2);
for i = 1:n2
    xre(i) = x(i);
    disp('The value of xre(i) is:'); disp( xre(i) );
end
for i = 0:n-1
    arg = twoPI * i / n;
    disp('The value of arg is:'); disp( arg );
end
```

While this is a strategy that can be manually performed for small-sized programs, using it for large-sized applications can be a tedious and unproductive task. Moreover, the code for tracing must also be manually removed from the final version of the code. In Section 3, we show how tracing can be achieved, in an elegant and systematic way, using the notion of program aspects.

2.2 Specialization of MATLAB Programs

The high level features of MATLAB make it a widely used language for fast development and for focusing on problem solving instead of on implementation issues. Type declarations, for example, do not exist in MATLAB.

The high level features, however, make it difficult to generate efficient implementation code from MATLAB programs. Also, and this is particularly true for embedded systems, it is commonly necessary to specify a particular MATLAB implementation to a target system, with hardware or computational restrains.

Let us consider the function `dft_custom` again. Transforming its code to a program in a different programming language used in target embedded systems can be challenging due to the dynamic nature of MATLAB's type system: type information is not explicit. To generate efficient code we do not only need type information, but we may also need different versions of the specialized function, one for each of the target systems. Next, we present the redefinition of `dft_custom`, where a single fixed point data type is explicitly defined by the programmer.

Example of a MATLAB Program with specialization:

```
function [mag] = dft_custom_specializing(x, n)
q = quantizer('fixed','floor', 'wrap', [32 16]);
```

```

twoPI = 2.0 * pi;
    twoPI = quantize(q, twoPI);
n2 = n/2;
    n2 = quantize(q, n2);
for i = 1:n2
    xre(i) = x(i);
    xre(i) = quantize( q, xre(i) );
end
for i = 0:n-1
    arg = twoPI * i / n;
    arg = quantize(q, arg);
end

```

In order to achieve this specialization, we created the MATLAB object `q`, defined using `quantizer` [5], which itself takes a series of arguments that determine the properties of `q`. The properties are used throughout the code to specialize the variables that are used to the data type whose properties are defined in `q`. In this case, we aim at targeting a generic system with variables in signed fixed-point mode (`fixed`), rounded towards negative infinity (`floor`), wrap on overflow (`wrap`) and has respectively 32 and 16 bits for the word length and for the fraction [2].

Regarding the code for specializing `dft_custom` that we presented above, it is as cumbersome as the code for tracing that we also present above. Indeed, it follows the same inefficient methodology that we have followed before. Again, the problem of data type and shape resolution and the generation of different implementations according to the target domain and architecture can be solved using aspect-oriented programming (AOP) [6].

As we discuss in the next section, our aspect language can be used to extend MATLAB programs with transformation and specialization rules that help the compiler to achieve more efficient code considering a certain target system. Furthermore, due to the modularity of our solution, the programs we obtain can easily be adapted to different deployment environments.

3 Aspect MATLAB

As mentioned in the previous section, the flexibility of the MATLAB language sometimes hinders performance and forces programmers to develop specialized versions of the same program. Furthermore, when it comes to evaluating specific features, such as tracing or including handlers to watch certain behaviors, the programmer is overwhelmed by cumbersome, error-prone and tedious tasks, which imply invasive code in the original MATLAB program. In [7], we have proposed aspect-oriented features to support triggering conditions and monitoring variable values, as well as a draft of an aspect language to support these features. In this paper, we briefly describe our aspect language specification, with primitives to create aspects, specify pointcut expressions, and apply transformations.

² Besides these options, MATLAB offers a wide range of possibilities to specialize types, as shown in [5].

The original base program is free of language enhancements and sources remain legal MATLAB. The proposed Domain Specific Aspects Language (DSAL) enables programmers to retain the obvious advantages of a single source program representation while allowing the implementations to explore a wide range of specific solutions at reduced programming and maintenance costs.

The template shown below illustrates the structure of an aspect module; the same file can have various aspect modules.

The structure of an aspect module:

```
aspect aspect_name
  select: Join Point Capture
  apply: Action Description :: execute before | after | around
end
```

Each aspect has a main constructor: `aspect`, which initializes the aspect and gives it a name, and two main sections: `select` and `apply`, where the join point and the action are declared. The section `apply` is followed by a primitive `execute` where, similarly to AspectJ [8], we define if the alteration to the join point occurs before, after, or around the join point, replacing the original code.

Table 1. Primitives for join point capture

Arrays	Variables/Constants	Functions
<code>add()</code>	<code>read()</code>	<code>call()</code>
<code>get()</code>	<code>write()</code>	<code>function()</code>
<code>size()</code>	<code>declare()</code>	<code>head()</code>

The join points capture works in functions, variables and arrays, and the functions that capture join points are given in Table 1. In the next sections, we show how concrete aspects may be defined to achieve the same tracing and specialization results that we have presented in Section 2.

3.1 Tracing

In Section 2.1, we have shown how to manually change the original `dft_custom` function to perform tracing. This was achieved by inserting obtrusive instructions in the original function definition. In this section, we present how to concisely specify such features using our aspect language, as specified in the next aspect.

The structure of an aspect to implement tracing:

```
aspect variable_tracing()
  select: write()
  apply(): "disp('The value of"++name++"is: ');disp("++name++");"
  :: execute after
end
```

First, we define an aspect with name `variable_tracing` that is responsible for tracing a variable. The join point `write` detects when a value is written to a variable. The `apply` primitive inserts the new code and the `execute` primitive demands its insertion to be after the join point.

Aspects are automatically woven to the original code in order to create a new, valid MATLAB program that has tracing capabilities, as shown in Section 2.1. This means that the changes performed by the programmer are easy to realize in practice and that, after they are used, it is easy to discard them. In fact, the original MATLAB function remains untouched during the entire process.

3.2 Function Specialization

MATLAB types are not mandatory: the dynamic type system allows the programmer to create variables and functions without specifying their types. When targeting MATLAB into a specific target, however, type information is crucial not only to produce efficient code but also to make the solution compatible with different processor architectures or other hardware requisites.

Specifying generic functions is easily done with our aspect oriented language for MATLAB and is very similar to the aspects shown in the previous section for tracing. The main difference here is that instead of only inserting information after the variables in order to force them to our custom types, we also have to define the `quantizer` object, that represents these types. Therefore, we now need two aspects: one that is executed only once, which is responsible for creating the `quantizer` object, and another that acts every time an assignment occurs, forcing that assignment into the desired type. These two aspects are presented next:

The structure of aspects to implement specialization:

```

aspect variable_specialization()
  select: head(dft_custom)
  apply(): "q = quantizer('fixed','floor', 'wrap', [32 16]);"
  :: execute after
end
aspect variable_specialization()
  select: write()
  apply(): name ++ "= quantize(q," ++ name ++ " );"
  :: execute after
end

```

The use of our aspect language makes it easier to specialize a MATLAB solution, but it also allows fast development and deployment of applications on heterogeneous environments where traditional programming techniques would not only be tedious and time-consuming but also prone to generate errors.

4 Metrics for MATLAB Programs

In the previous sections, we introduced both MATLAB and an aspect oriented extension for it. In this section, we briefly present complexity metrics for MATLAB

and we introduce aspect MATLAB metrics. The idea is to use the complexity metrics to assess the quality of a MATLAB program and compare it to its aspect version.

4.1 Complexity Metrics for MATLAB

To assess the complexity of MATLAB programs, we use the *lines of code* metric and the *Halstead's* complexity metric suite [3].

Lines of Code (LOC): this metric is frequently used in software engineering to assess the quality of source code. Through it one can predict the effort needed to create the program and the cost of maintaining it after it is produced.

Halstead's Complexity: this suite of metrics was developed to measure a program's complexity directly from source code. The suite is composed by six measures that emphasize the complexity of a program: *Program Vocabulary*, *Program Length*, *Calculated Program Length*, *Volume*, *Difficulty*, and *Effort*.

Despite being easy to calculate, in order to automate the measuring process we have to define strong rules for identifying the operands and operators [9]. Let n_1 , n_2 , N_1 , and N_2 be the number of distinct operators, the number of distinct operands, the total number of operators, and the total number of operands. The metrics that constitute the Halstead suite are, then, defined as follows:

- **Program Vocabulary (VOC):** $n = n_1 + n_2$
- **Program Length (PL) :** $N = N_1 + N_2$
- **Calculated Program Length (CPL):** $\hat{N} = n_1 \times \log_2 n_1 + n_2 \times \log_2 n_2$
- **Volume:** $V = N \times \log_2 n$
- **Difficulty:** $D = \frac{n_1}{2} \times \frac{N_2}{n_2}$
- **Effort:** $E = D \times V$

It is important to notice that these results provided by Halstead are not very useful by themselves, since they do not have units and therefore there is no qualitative interpretation for them (which is well-known problem of the Halstead suite). Consequently, we apply them in the basis that they are useful only when directly comparing different MATLAB sources.

4.2 Metrics for Aspects in MATLAB

Together with the metrics for MATLAB programs, we introduce a suite of metrics to help measuring the impact and the benefits of using an aspects oriented language when programming in MATLAB. For this we adapt the four software metrics introduced by [10] and [11] to MATLAB.

Concern Diffusion over Lines of Code (CDLOC): this metric counts the number of transition points, in the source, in and out of zones where a concern starts and ends (shadowed zones). The use of this metric requires a shadowing process that partitions the code into shadowed areas and non-shadowed areas,

being the code inside the shadowed areas lines of code that implements a concern. Transition points are points in the code where there is a transition from a non-shadowed to a shadowed area and vice-versa [11].

Tangling Ratio (TR): this metric gives an estimation about tangling on the program source code [10]. In the context of MATLAB, we can define it using the following formula:

$$\text{Tangling Ratio} = \frac{\text{CDLOC program}}{\text{LOC program}}$$

Concern Impact on LOC (CILOC): this metric gives us the ratio between a original MATLAB source code free of concerns, and the code after being transformed by our aspect language. This metric allows us to have a first intuition about the impact of using aspects in terms of lines of code, and it is given by the formula:

$$\text{Concern Impact LOC} = \frac{\text{LOC of concerns free program}}{\text{LOC of transformed program}}$$

The range of the results ranges from zero to one, where one means there are no concerns on the MATLAB solution and, therefore, there is no advantage in using an aspect oriented language. As we will see though, this is very uncommon since concerns are usually an important part of any software solution, being it MATLAB or not.

Aspectual Bloat (AB): measures the aspects in terms of LOC bloat in the MATLAB programs [10]. It is calculated by the following formula:

$$\text{AspectualBloat} = \frac{\text{LOC with concerns} - \text{LOC without concerns}}{\text{LOC of aspects}}$$

When the result of this metric is 1, it means that the number of lines written for the aspects plus the number of lines written on the MATLAB program without aspects is equal to the number of lines of the MATLAB program with aspect oriented language. With this result, it might seem that there is no advantage in using an aspects language, but even if the effort, in terms of lines of code was the same, the use of an aspects oriented language has other advantages, such as creating a program which is more modular and, consequently, easier to maintain and update.

5 Metrics Evaluation

With the metrics presented in the previous section, we extended our MATLAB front-end (which uses the MATLAB to Tom-IR tool [12]) in order to be able to apply the metrics to both the original MATLAB source code and its aspect oriented variants.

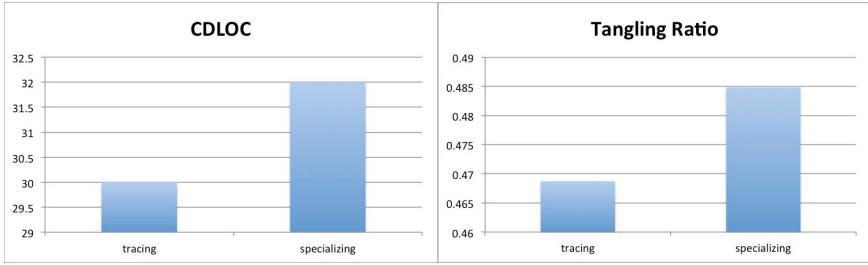


Fig. 1. CDLOC and Tangling Ratio on the manually transformed versions of `dft_custom`

5.1 Computing Metrics for Aspect MATLAB

In order to present our metrics we will use the examples provided in Section 2, consisting of both versions of function `dft_custom`: the one with variable tracing, as shown in Section 2.1 and the one with variable specialization, as shown in Section 2.2. Before, we had manually and intrusively written the code responsible for tracing and specialization. After that, we were able to run the first two metrics presented in Section 4.2, CDLOC and TR. These two metrics give us an indication of how many concerns exist in the source code and how much impact they have in the overall code quality. The results are presented in Figure 1.

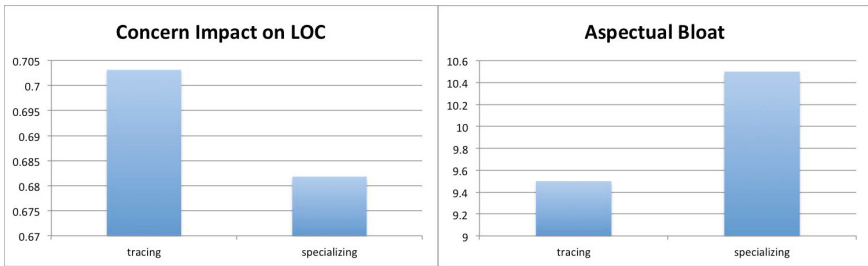


Fig. 2. Concern Impact and Aspectual Bloat on the aspect-oriented versions of `dft_custom`

We can see that both versions of the function `dft_custom` would benefit from the usage of an aspect language. Indeed, they have around thirty transition points in their code (as shown in the left chart of Figure 1), between a concern and the functional code of the application, and each transition point has the potential to be transformed into an aspect. Here, as it is often the case, we do have various concerns, and therefore transition points handled by a single aspect makes their usage even more valuable. Regarding TR (right chart of Figure 1), both functions also show clear signs of a high degree of code tangling.

A significant amount of concerns in the source code makes it harder to understand and maintain. The fact that the metrics achieved such expressive results for `dft_custom` shows a good potential for aspects to be applied: they are particularly good in modularizing and aiding on implementing such features.

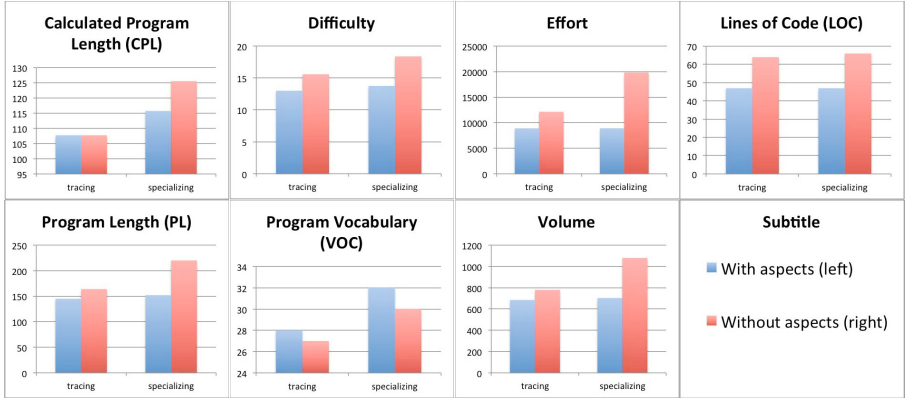


Fig. 3. Metrics for `dft_custom` with tracing and specialization, implemented with and without our aspects language

Our next step was to write, using our aspect language, modules that automatically generate the traced and specialized versions. These aspects are similar to the ones presented in Section 3, and make it not only simple and easy to obtain the different `dft_custom` functions, but also to backtrack any transformation in case we want to revert their application. In fact, the traditional method to backtrack a manually transformed function is to manually remove all the code that implements the aspect functionality.

In Figure 2 we show, through the use of metrics CILOC and AB the actual effect that using AOP had on function `dft_custom`.

The first metric results, presented in the left chart of Figure 2, show the relation between the lines on a version of the code without concerns and on the same code after being transformed by our aspect language. This shows how less effort is needed by using an aspect language. In this case, and particularly on the case with tracing, the effect was noticeable: we were able to inject a significant number of lines of code only through the use of a single aspect.

The second metric results, shown in the right chart of Figure 2, seem promising too. This metric shows how much aspect code we had to write to change the original source code. The observed results indicate that the programming effort was greatly reduced by using aspects.

5.2 Computing MATLAB Metrics

So far, we have shown that the use of aspects can help on implementing new features while minimizing the traditional negative impact of extending source

code. In this section we try to assess whether there are quality improvements in the original program, with added aspects, when compared to the manually transformed versions. For this, we compute the metrics presented in Section 4.1 on all versions of the `dft_custom` function, whose results are presented in Figure 3.

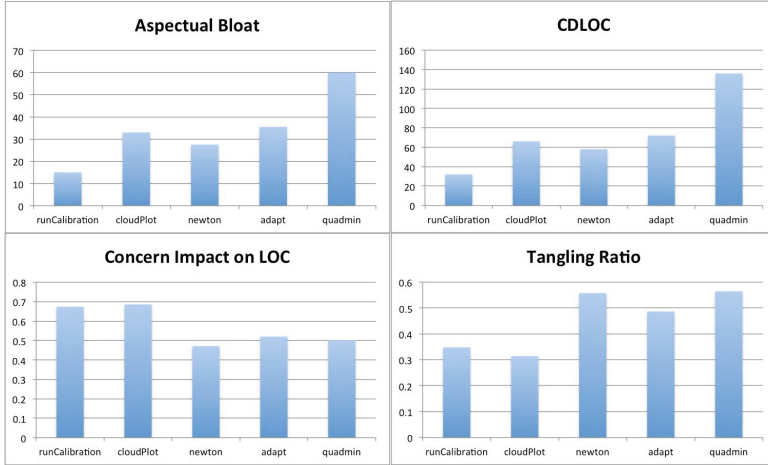


Fig. 4. Aspect metrics in a set of five MATLAB functions

This metrics suite seems to provide strong evidence that using aspects increases the overall code quality. The version with aspects is always clearer, shorter and easier to read. Some results actually show a large improvement, such as the metric *Difficulty*, that shows improvements in the order of 89% and *VOC*, that shows gaps in the order of 88%. Other metrics show smaller advantages, such as *CPL* that shows improvements in the order of 8%. Still, the overall analysis of all results leads us to believe that using aspects in MATLAB programs can create programs with better quality.

5.3 Quality Analysis in Real-Life Applications

To further test the impact of our aspects language, we used a set of five functions taken randomly from MATLAB's File Exchange [13]. This website is a community-based repository for MATLAB functions, applications and scripts, where users can upload their implementations and download programs.

For each application, we took one source code file (very commonly, a MATLAB solution is made out of various source files) while being careful to pick applications from the groups with best rating or with the highest number of downloads or comments. We did so to ensure that our set is actually representative of the code usually found in MATLAB.

Next, we manually inserted new features in the code in order to specialize it and to trace its variables, similarly to the `dft_custom` examples presented in Section 2. We also implemented these features using our aspects language, i.e., with aspects that transform the MATLAB sources into new versions that support tracing and specialization.

In Figure 4 we show the aspect metrics applied to this set of functions. The results are similar to the ones found for `dft_custom`: using aspects reduces the effort of implementing specialized versions as seen, for example, on the tangled concerns on the code. In Figure 5 we show the metrics used for MATLAB quality assessment. Again, the overall results prove that using aspects decreases the size of the solution while keeping the code easier to maintain and reducing the effort to understanding it. Some metrics are particularly positive, with improvements of around 90%, from the traditional implementation to the aspects oriented version, as for PL on the `quadadmin` function with tracing. The *Effort* metric applied to the same function also shows results on the order of 64%. The charts in Figure 5 confirm the overall promising results of our aspect oriented approach.

5.4 Quality Analysis of the Matlab Program IMPACTED

In order to further confirm the results obtained so far, we have applied our metrics suite to the MATLAB program IMPACTED [14,15,16]. This program focuses on hazard avoidance techniques for controlled landings and has been the object of several studies. It represents a mature, highly complex solution composed by various functions and gives us a good environment for testing MATLAB code regarding both code quality and the potential for AOP introduction.

We have chosen a subset of this package, representing approximately 270 lines of code obtained through profiling, which gave us the hotspot in terms of the execution, represented by ten functions that represent the most important computations together and highest overheads. We did so because typically it is not necessary to separate concerns on all the elements of a solution. Some elements represent auxiliary functions, simpler and easier to control. The core computations, on the other hand, represent the core functionalities and therefore the parts where errors are more crucial, and where code control is harder.

We followed the same analysis strategy we showed throughout Section 5, first analyzing the potential for aspects implementation, and secondly analyzing the changes on the final solutions improved with tracing and specialization when aspects were added to the programming cycle. Figure 6 and Figure 7 show the results of these two analysis, respectively.

The results show an overall improvement in code quality, increasing as much as 21% in LOC and PL. The effort to understand the code shows improvements of 26% and 52% for the versions of IMPACTED where tracing and specializing were introduced. On other charts we see worse results, such as the CPL that

shows no improvements at all in the version with tracing and only 37% for the specialized version. A metric in particular, the *Vocabulary*, shows an increase in the implementing effort. This might be due the fact that, by introducing a new language, we are forcing the programmer to learn a new set of constructors and primitives. We do not see this as a pitfall though, as it is a one of task and the overall improvements in all the other metrics prove it is worth it.

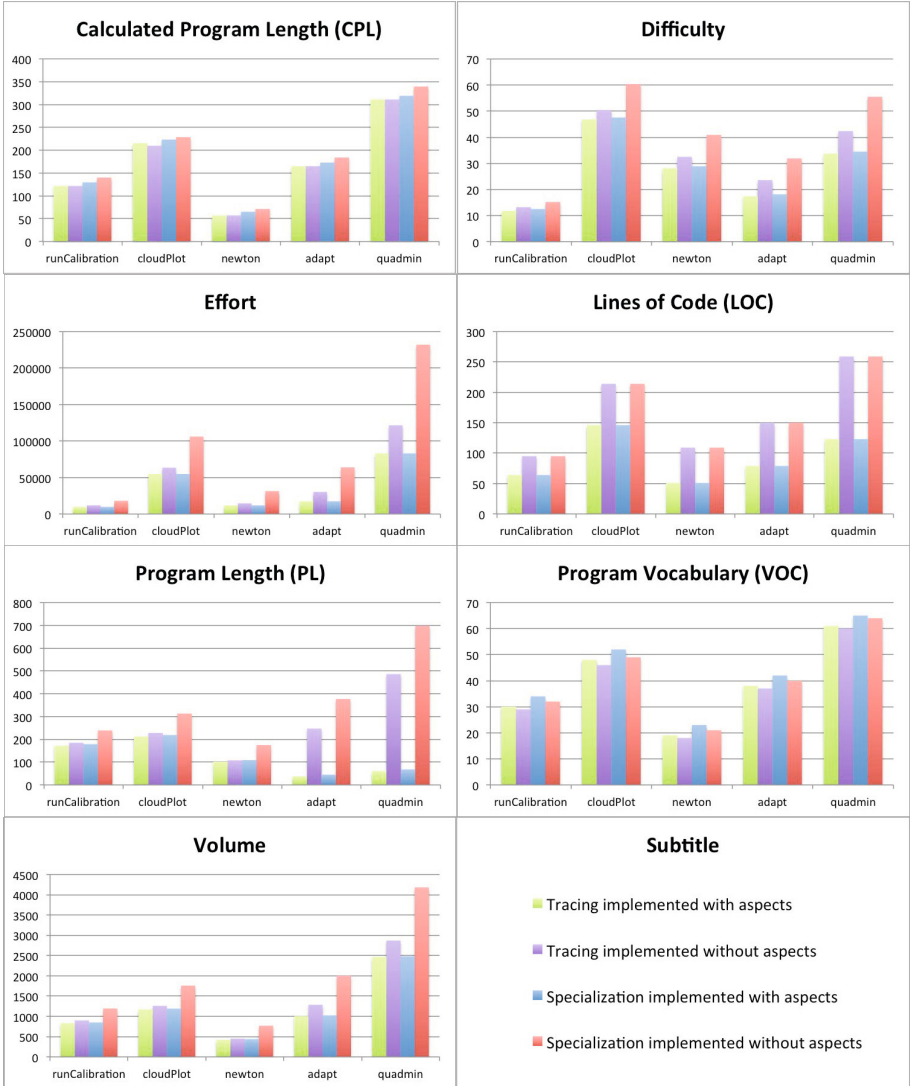


Fig. 5. Metrics results for each function with tracing and specialization, implemented with and without aspects

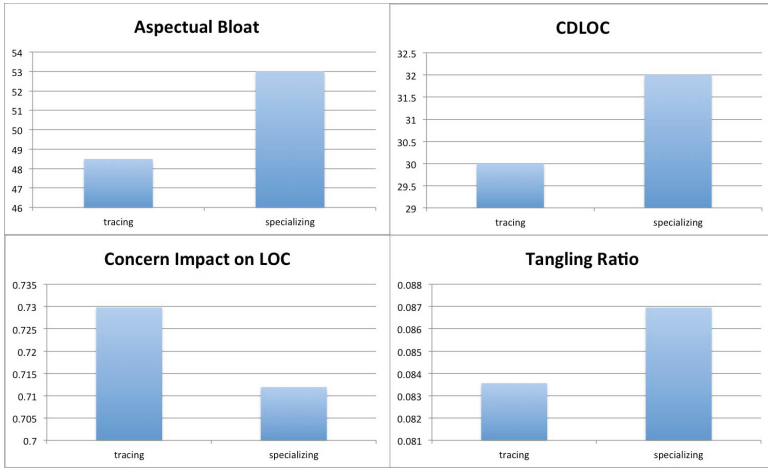


Fig. 6. Aspect metrics in IMPACTED

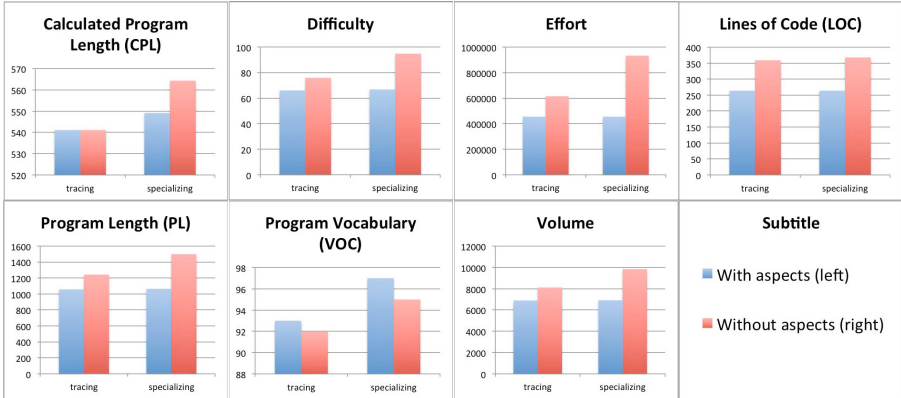


Fig. 7. Metrics for IMPACTED with tracing and specialization, implemented with and without our aspects language

6 Implementation

The metrics presented are implemented in our MATLAB front-end. This front-end includes parsers and construction of the abstract syntax tree for MATLAB (with the MATLAB to Tom-IR tool [12]) and for the MATLAB aspect language.

This front-end was developed using advanced language engineering techniques, like generalised (top-down) parsing (using the ANTLR parser generator [17]), strategic programming [18,19,20] (implemented with the TOM system [21]), attribute grammars [22,23] (implemented in the Lrc system [24]), and formal program calculation techniques to reason about our implementations [25,26]. By using these we can easily define tree-traversal algorithms, that we heavily use to weave the abstract data-types of both the aspects and the MATLAB code.

These techniques allowed us to implement all metrics presented in this paper in a concise and generic way. That is, they are independent of the abstract tree. As a consequence, new metrics can be easily added to our metric suite.

7 Conclusion

In this paper we presented software metrics for assessing the software complexity of both standard MATLAB programs and aspect oriented MATLAB programs.

We adapted a set of AOP metrics to the Aspect MATLAB realm, by implementing them in our MATLAB front-end, and used them to assess the complexity of MATLAB programs when compared to their AOP equivalents. We did so by applying our suite first to a set of widely used MATLAB functions and later to a fully developed MATLAB program, consisting of many MATLAB functions.

Our preliminary results are promising and show that aspect oriented programming in MATLAB improves the quality of programs. Both the MATLAB metrics and the aspect metrics are implemented in our (Aspect) MATLAB frontend.

References

1. MATLAB: version 7.10.0 (R2010a). The MathWorks Inc., Natick, Massachusetts (2010)
2. MathWorks: Front page, <http://www.mathworks.com> (accessed in February 2012)
3. Halstead, M.H.: Elements of Software Science. Operating and programming systems series. Elsevier Science Inc., New York (1977)
4. Aslam, T., Doherty, J., Dubrau, A., Hendren, L.: Aspectmatlab: an aspect-oriented scientific programming language. In: Proceedings of the 9th International Conference on Aspect-Oriented Software Development (AOSD), pp. 181–192. ACM, New York (2010)
5. MathWorks: R2012a documentation - fixed-point toolbox, <http://www.mathworks.com/help/toolbox/fixedpoint/ref/quantizer.html> (accessed in February 2012)
6. Cardoso, J., Fernandes, J., Monteiro, M.: Adding aspect-oriented features to matlab. In: Workshop on Software Engineering Properties of Languages and Aspect Technologies (SPLAT! 2006) (2006)
7. Cardoso, J., Diniz, P., Monteiro, M.P., Fernandes, J.M., Saraiva, J.: A domain-specific aspect language for transforming MATLAB programs. In: Fifth Workshop on Domain-Specific Aspect Languages (DSAL) (March 2010)
8. Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W.G.: An Overview of AspectJ. In: Lee, S.H. (ed.) ECOOP 2001. LNCS, vol. 2072, pp. 327–353. Springer, Heidelberg (2001)
9. Peckhan, J., Lloyd, S.J.: Practicing Software Engineering in 21st century. IRM Press (2003)
10. Lopes, C.V.: D: A Language Framework for Distributed Programming. PhD thesis, College of Computer Science, Northeastern University (1997)
11. Sant'anna, C., Garcia, A., Chavez, C., Lucena, C., v. von Staa, A.: On the reuse and maintenance of aspect-oriented software: An assessment framework. In: Proceedings XVII Brazilian Symposium on Software Engineering (SBES) (2003)

12. Nobre, R., Cardoso, J.M.P., Diniz, P.C.: Leveraging type knowledge for efficient matlab to c translation. In: 15th Workshop on Compilers for Parallel Computing (CPC) (2010)
13. MathWorks: Matlab central - file exchange, <http://www.mathworks.com/matlabcentral/fileexchange> (accessed in February 2012)
14. Devouassoux, J., Reynaud, S., Jonniaux, G., Ribeiro, R.A., Pais, T.C.: Hazard avoidance developments for planetary exploration. In: 7th International ESA Conference on Guidance, Navigation and Control Systems (2008)
15. Reynaud, S., Drieux, M., Bourdarias, C., Philippe, C., Pham, B.v., Astrium Space Transportation: Science driven autonomous navigation for safe planetary pin-point landing 1. Context, 1–10 (2009)
16. Pais, T., Ribeiro, R.A.: Contributions to dynamic multicriteria decision making models. In: Proceedings of the International Fuzzy Systems Association World Congress and European Society for Fuzzy logic and technology Conference (IFSA-EUSFLAT), pp. 719–724 (2009)
17. Parr, T.: The Definitive ANTLR Reference: Building Domain-Specific Languages, 1st edn. Pragmatic Programmers. Pragmatic Bookshelf (2007)
18. Visser, J., Saraiva, J.: Tutorial on strategic programming across programming paradigms. In: 8th Brazilian Symposium on Programming Languages, SBLP (2004)
19. Balland, E., Moreau, P.E., Reilles, A.: Rewriting strategies in java. *Electron. Notes Theor. Comput. Sci.* 219, 97–111 (2008)
20. Visser, E.: Program Transformation with Stratego/XT: Rules, Strategies, Tools, and Systems in Strategoxt-0.9. In: Lengauer, C., Batory, D., Blum, A., Vetta, A. (eds.) *Domain-Specific Program Generation*. LNCS, vol. 3016, pp. 216–238. Springer, Heidelberg (2004)
21. Balland, E., Brauner, P., Kopetz, R., Moreau, P.-E., Reilles, A.: Tom: Piggybacking Rewriting on Java. In: Baader, F. (ed.) *RTA 2007*. LNCS, vol. 4533, pp. 36–47. Springer, Heidelberg (2007)
22. Knuth, D.E.: *Semantics of Context-free Languages*. *Mathematical Systems Theory* 2(2), 127–145 (1968); Correction: *Mathematical Systems Theory* 5(1), 95–96 (March 1971)
23. Saraiva, J., Swierstra, D.: Generating Spreadsheet-Like Tools from Strong Attribute Grammars. In: Pfenning, F., Smaradakis, Y. (eds.) *GPCE 2003*. LNCS, vol. 2830, pp. 307–323. Springer, Heidelberg (2003)
24. Kuiper, M., Saraiva, J.: Lrc - A Generator for Incremental Language-Oriented Tools. In: Koskimies, K. (ed.) *CC 1998*. LNCS, vol. 1383, pp. 298–301. Springer, Heidelberg (1998)
25. Fernandes, J.P., Pardo, A., Saraiva, J.: A shortcut fusion rule for circular program calculation. In: *ACM SIGPLAN Haskell Workshop, Haskell 2007*, pp. 95–106. ACM, New York (2007)
26. Pardo, A., Fernandes, J.P., Saraiva, J.: Shortcut fusion rules for the derivation of circular and higher-order programs. In: *Higher-Order and Symbolic Computation*, pp. 1–35. Springer (2011)

A Suite of Cognitive Complexity Metrics

Sanjay Misra¹, Murat Koyuncu¹, Marco Crasso²,
Cristian Mateos², and Alejandro Zunino²

¹ Department of Computer Engineering, Atilim University, Ankara, Turkey
{smisra, mkoyuncu}@atilim.edu.tr

² ISISTAN Research Institute. UNICEN University, Tandil, Argentina. Also Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET).
{mcrasso, cmateos, azunino}@conicet.gov.ar

Abstract. In this paper, we propose a suite of cognitive metrics for evaluating complexity of object-oriented (OO) codes. The proposed metric suite evaluates several important features of OO languages. Specifically, the proposed metrics are to measure method complexity, message complexity (coupling), attributes complexity and class complexity. We propose also a code complexity by considering the complexity due to inheritance for the whole system. All these proposed metrics (except attribute complexity) use the cognitive aspect of the code in terms of cognitive weight. All the metrics have critically examined through theoretical and empirical validation processes.

Keywords: software metrics, methods, messages, attributes, class, coupling, inheritance, cognitive complexity, validation.

1 Introduction

IEEE defines the software quality as ‘Software quality is the degree to which software possesses a desired combination of attributes (e.g., reliability, interoperability)’ [1]. Software quality is controlled by software metrics. Software metrics are tools to control the complexity of software. Through metrics, one can easily observe the several weaknesses of a software system and therefore, by means of it, quality can be estimated. This is the reason why metrics are indispensable tool in software development life cycle for achieving the quality.

The recent decades have witnessed the successes of the object-oriented (OO) languages. Most of the projects are being developed in JAVA, C++ or in Python. The need to control the complexity of the projects developed in these language is important. For this purpose, since the beginning of the 1990 several object-oriented metrics e.g. Chidamber and Kemerer (CK) metrics suite [2], MOOD metrics for OO Design [3], design metrics for testing [4], product metrics for object-oriented design [5-6], Lorenz and Kidd metrics [7], Henderson–Seller metrics [8], (slightly) modified CK metrics [9], size estimation of OO systems[10], weighted class complexity metric [11] and several other metrics[12-17] can be found in the literature. All the above metrics tried to cover some features of the OO languages and used for some quality

attributes. The quality attributes, are such as correctness, reliability, efficiency, integrity, usability, maintainability, testability, flexibility, portability, reusability, and interoperability [18]. Amongst the given quality attributes, maintainability is treated as the most necessary attribute for software products [19]. In fact, majority of the metrics are developed for this most important attribute.

In our previous work, we have presented metrics for OO codes [11]. For inheritance complexity, we have first calculated the cognitive weights of all the methods of a class, sum up them and then multiply with the weight of their parent classes (due to inheritance). However, later, we have observed that while considering complexity due to inheritance, we should not only consider the method complexity but the complexity due to attributes. In this point of view, while estimating the complexity of the entire OO codes, we have to calculate the complexity of a class by considering the impact due to method complexity, message complexity and also due to attributes [11]. Then, we have to establish a relation between classes to capture the complexity due to inheritance property. The mentioned requirement is the starting point of this work and we present a suite of metrics which capture most of features of the OO programming paradigm in this paper.

The paper is organized as follows: The motivation of the work is given in the next section. Section 3 presents the proposal of the new suite of complexity metrics. The metrics are demonstrated with an OO example in Section 4. Finally, a conclusion is given in Section 5.

2 Motivation

After the CK metric suite, no further attempts have been made seriously in this direction to develop a more effective suite of metrics for OO languages [2]. All the metrics in CK metric suite are straight forward and simple to compute. On the other hand, these metrics do not cover the following issues:

1. The overall complexity of a class due to all possible factors
2. The internal architecture of the class
3. The impact of the relationship due to inheritance in the class hierarchy
4. The number of messages between classes and their complexities (CK metrics suite counts only the methods coupled with other classes)
5. Cognitive complexity, which is a measure of understandability and therefore has a great impact on maintainability of the system

The lack of the above features in CK metric suite motivated us to produce a new suite of metrics, which can be a complimentary set of the CK metric suite. In fact, our proposed metrics suggest examining the OO properties in more detail. For example, CBO (one of metrics in CK metric suite) is a measure to show interactions between objects by counting the number of other classes to which the class is coupled. In our proposal, coupling is computed by considering the message calls to other classes and

the weight of the called methods. One class may have “1” for CBO showing that it interacts with only one class, but may include many messages to that class which causes a more complex code (which is considered in our metric). Therefore, we believe that our metric gives more accurate information about coupling of a class.

3 Proposed Suite of Metrics for Object-Oriented Programming

An object is a class instance and an object-oriented system consists of objects which collaborate through message exchanges. An object-oriented code includes one or more classes which may be related to each other by composition or inheritance and contains related attributes and operations (methods) in the classes. The complexity metrics developed for object-oriented languages are mainly based on the complexity of individual classes like number of methods, number of messages etc. However, not only the numbers of different components are important, but also the internal complexities of these components are equally important. Furthermore, for calculating the complexity of the entire system, we have to consider the special features of OO programs and type of the relations between classes. Accordingly, we propose the following suite of metrics:

Method Complexity(MC): Method complexity is calculated by considering corresponding cognitive weights of structures in a method of a class. Cognitive weights are used to measure the complexity of the logical structures of the software in terms of Basic Control Structures (BCSs). These logical structures reside in the method (code) and are classified as sequence, branch, iteration and call with the corresponding weights of one, two, three and two, respectively. Actually, these weights are assigned on the classification of cognitive phenomenon as discussed by Wang [20]. We calculate method complexity in a class by associating a number (weight) with each member function (method), and then simply add the weights of all the methods. More formally, the method complexity (MC) is calculated as;

$$MC = \sum_{j=1}^q \left[\prod_{k=1}^m \sum_{i=1}^n W_c(j, k, i) \right], \quad (1)$$

where, W_c is the cognitive weight of the concerned Basic Control Structure (BCS). The method complexity of a software component is defined as the sum of cognitive weights of its q linear blocks composed of individual BCSs, since each block may consist of m layers of nested BCSs, and each layer with n linear BCSs. Equation 1 gives the complexity of a single method.

Message complexity (Coupling Weight for a Class (CWC)): Two classes are coupled when there is a message call in one class for the other class. In our proposal, if there are message calls for other classes, we not only count the total number of such messages, but also we add the weight of the called methods. Accordingly,

complexities due to message calls are the sum of weights of call and the weight of called methods. i.e.

$$CWC = \sum_{i=1}^n (2 + MC_i), \quad (2)$$

where, 2 is the weight of the message to an external method and W_i is the weight of the called method. If there are n numbers of external calls, then the CWC is calculated as the sum of weights of all message calls.

Attribute Complexity(AC): It reflects the complexity due to data members (attributes). We simply assign the total number of attributes associated with class as the complexity due to data members. The attributes are not local to one procedure but local to objects and can be accessed by several procedures. Accordingly, the attribute complexity of a class (AC) is given by:

$$AC = \sum_{i=1}^n 1, \quad (3)$$

where n is the total number of attributes.

Weighted Class Complexity(WCC): OO software development is based on classes and subclasses whose elements are attributes and methods (including messages). These elements are identified in class declarations and are responsible for the complexity of a class. Therefore, the complexity of a class is a function of the methods and the data attributes. More formally, we suggest the following formula to calculate the Weighted Class Complexity (WCC):

$$WCC = AC + \sum_{p=1}^n MC_p \quad (4)$$

WCC is the sum of the attribute complexity and the sum of all the method complexities of a class.

Code Complexity (Inheritance): For calculating the complexity of the entire system, we have to consider not only the complexity of all the classes, but also the relations among them. That is, we are giving emphasis on the inheritance property because classes may be either parent or children classes of others. In the case of a child class, it inherits the features from the parent class. By keeping this property of OO paradigm, we propose to calculate the code complexity of an entire system as follows:

- If the classes are of the same level then their weights are added.
- If they are subclasses or children of their parent then their weights are multiplied.

If there are m levels of depth in the object-oriented code and level j has n classes then the Code Complexity (CC) of the system is given by,

$$CC = \prod_{j=1}^m \left[\sum_{k=1}^n WCC_{jk} \right] \quad (5)$$

The unit of CC is defined as the cognitive weight of the simplest software component (having a single class which includes single method having only a sequential structure). This corresponds to sequential structure in BCS and hence its unit is taken as 1 Code Complexity Unit (CCU).

In addition to these metrics, we are also proposing the associated metrics which are extracted from the above metrics. These metrics may be useful indications for general information regarding the projects.

Average Method Complexity(AMC): It gives an average method complexity for a class and is calculated by dividing the sum of complexities of all the methods of a class to the total number of methods in that class.

$$AMC = \sum_{p=1}^n MC_p / n, \quad (6)$$

where MC is the complexity of a particular method, n is total number of methods in a class.

Average Method Complexity per Class(AMCC): It is defined as the average method complexity for the entire system.

$$AMCC = \sum_{p=1}^m AMC / m, \quad (7)$$

where m is total number of classes in a project.

Average Class Complexity(ACC): It is the average complexity of classes in a project and it is calculated by dividing the sum of the complexity of the classes to the total number of classes.

$$ACC = \sum_{p=1}^m WCC / m, \quad (8)$$

where WCC is the complexity a class and m is total number of classes.

Average Coupling Factor(ACF): It is defined as the complexity of all the external method calls (i.e. coupling weights) to the total number of messages.

$$ACF = \sum_{i=1}^k CWC / k \quad (9)$$

where k is number of messages to other classes.

Average Attributes per Class(AAC): It shows the average number of attributes per class in a project and it is calculated by dividing the sum of attribute complexity of all classes to the total number of classes.

$$AAC = \sum_{i=1}^m AC / m, \quad (10)$$

where, m is the total number of classes.

4 Demonstration of the Metrics

The proposed complexity metrics given in Section 3 is demonstrated with a programming example in this section. The class hierarchy of the example program is illustrated in Fig.1 and the complete C++ code for the example is given in Appendix.

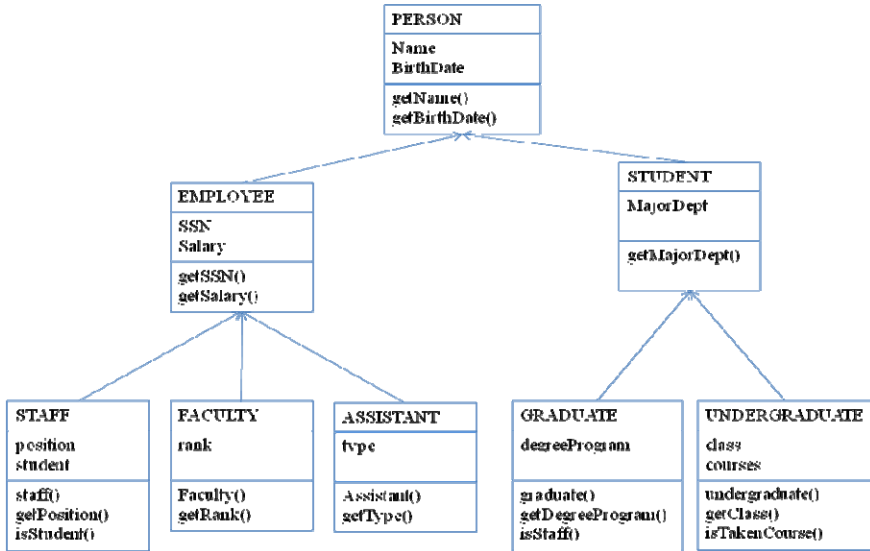


Fig. 1. An Example Class Hierarchy

The given example processes a personnel database hierarchy. It has one main class *Person* and two subclasses, *Employee* and *Student*. The class *Employee* has again three subclasses, *Staff*, *Faculty*, and *Assistant*. The *student* class has two subclasses, *Graduate* and *Undergraduate*. This section demonstrates how we calculate the complexities according to the metrics given in section 3 for an object-oriented program.

Method Complexity (MC): Method complexity of each method is calculated using Formula 1. For example, the class *Person* has two methods named as *getName()* and *getBirthDate()*. Each of these methods has simple structure, called as sequence basic control structure (BCS), therefore their weights are assigned as 1.

$$MC_{getName}=MC_{getBirthDate}=\sum_{j=1}^q \left[\prod_{k=1}^m \sum_{i=1}^n W_c(j, k, i) \right] = 1$$

So, the method complexity for the class person:

$$MC_{PERSON}=MC_{getName} + MC_{getBirthDate} = 1+1=2 \text{ CCU}$$

For another example, consider *isStudent()* method of the *Staff* class. The method includes an IF structure (branch). The method complexity is calculated as:

$MC_{isStudent} = 1+2 = 3$, where 1 is for sequence and 2 is for branch (IF) structure.

The method *isStaff()* of the class *Graduate* shows a more detailed example:

$MC_{isStaff} = 1+2((2+3)+2) = 15$, where 1 is for sequence and 2 is for branch (IF) structure. The branch structure has an external method call and a nested branch inside. $(2+3)$ is for the message sent to another class (i.e., *Staff*), 2 is the weight of the message and 3 is the weight of the called method (i.e. *isStudent()*). The last 2 in the calculation is for the nested IF structure. Notice that if there is nested structure, we multiply the weights instead of summing them up.

The method complexity for the classes given in Fig.1 is calculated as follows:

$$MC_{PERSON} = MC_{getName} + MC_{getBirthDate} = 1+1=2 \text{ CCU}$$

$$MC_{EMPLOYEE} = MC_{getSalary} + MC_{getSSN} = 1+1=2 \text{ CCU}$$

$$MC_{STUDENT} = MC_{getMajorDept} = 1 \text{ CCU}$$

$$MC_{STAFF} = MC_{staff} + MC_{getPosition} + MC_{isStaff} = 1+1+3=5 \text{ CCU}$$

$$MC_{FACULTY} = MC_{faculty} + MC_{getRank} = 1+1=2 \text{ CCU}$$

$$MC_{ASSISTANT} = MC_{assistant} + MC_{getType} = 1+3=4 \text{ CCU}$$

$$MC_{GRADUATE} = MC_{graduate} + MC_{getDegreeProgram} + MC_{isStaff} = 1+1+15=17 \text{ CCU}$$

$$MC_{UNDERGRADUATE} = MC_{undergraduate} + MC_{isTakenCourse} + MC_{getClass} = 4+7+1=12 \text{ CCU}$$

All the method complexities can be seen in Appendix along with the code of each method.

Message complexity (Coupling Weight for a Class (CWC)): In the given example, there is only one class (*Graduate*) which includes one external message call to the *Staff* class. We can calculate the coupling weight of the class *Graduate* as the weight of the called methods. For this example, there is only one external method called from the *Graduate* class (*isStaff()*).

$$CWC = \sum_{i=1}^n (2 + MC_i) = 2 + 3 = 5 \text{ CCU}$$

Attribute Complexity (AC): Attribute complexity of a class can be calculated by counting the total number of attributes in that class. Accordingly, AC values for classes Person, Employee, Student, Faculty, Staff, Assistant, Graduate and Undergraduate are 2, 2, 1, 2, 1, 1, 1 and 2, respectively.

Weighted Class Complexity (WCC): WCC for each class can be calculated by Equation 4, i.e the sum of attribute complexity, method complexity and message complexity. It is worth mention that while calculating the WCC, we don't need to include the message complexity of classes, because, a message is a part of a method and already calculated in the method complexity. For the given example, WCCs are calculated as follows:

$$\begin{aligned}
WCC_{\text{PERSON}} &= 2+2= 4 \text{ CCU} \\
WCC_{\text{EMPLOYEE}} &= 2+2= 4 \text{ CCU} \\
WCC_{\text{STUDENT}} &= 1+1= 2 \text{ CCU} \\
WCC_{\text{STAFF}} &= 2+5= 7 \text{ CCU} \\
WCC_{\text{FACULTY}} &= 1+2= 3 \text{ CCU} \\
WCC_{\text{ASSISTANT}} &= 1+4= 5 \text{ CCU} \\
WCC_{\text{GRADUATE}} &= 1+17= 18 \text{ CCU} \\
WCC_{\text{UNDERGRADUATE}} &= 2+12= 14 \text{ CCU}
\end{aligned}$$

Code Complexity(CC): The code complexity of the object-oriented code is calculated by using Equation 5 as given below:

$$\begin{aligned}
CC &= WCC_{\text{PERSON}} * (WCC_{\text{EMPLOYEE}} *(WCC_{\text{STAFF}} + \\
&\quad WCC_{\text{FACULTY}} + WCC_{\text{ASSISTANT}})+ WCC_{\text{STUDENT}} * \\
&\quad (WCC_{\text{GRADUATE}} + WCC_{\text{UNDERGRADUATE}})) \\
&= 4*(4*(7+3+5) + 2*(18+14)) \\
&= 496 \text{ CCU}
\end{aligned}$$

Average Method Complexity(AMC):

$$AMC = \sum_{p=1}^n MC_p / n, \text{ where } W \text{ is the weight of a particular method, } n \text{ is}$$

total number of method in a class.

$$\begin{aligned}
AMC_{\text{PERSON}} &= 2/2= 1 \text{ CCU} \\
AMC_{\text{EMPLOYEE}} &= 2/2= 1 \text{ CCU} \\
AMC_{\text{STUDENT}} &= 1/1= 1 \text{ CCU} \\
AMC_{\text{STAFF}} &= 5/3= 1.66 \text{ CCU} \\
AMC_{\text{FACULTY}} &= 2/2= 1 \text{ CCU} \\
AMC_{\text{ASSISTANT}} &= 4/2= 2 \text{ CCU} \\
AMC_{\text{GRADUATE}} &= 17/3= 5.66 \text{ CCU} \\
AMC_{\text{UNDERGRADUATE}} &= 12/3= 4 \text{ CCU}
\end{aligned}$$

Average Method Complexity per Class(AMCC):

$$AMCC = \sum_{p=1}^m AMC / m, \text{ where } m \text{ is total number of classes in a project}$$

$$AMCC = (1 + 1 + 1 + 1.66 + 1 + 2 + 5.66 + 4) / 8 = 2.165 \text{ CCU}$$

The average method complexity of a class is 2.165. It is worth mentioning that this number is not the average number of methods per class but it represents the average complexity/weight of method per class

Average Class Complexity(ACC):

$$ACC = \sum_{p=1}^m WCC / m, \text{ where } WCC \text{ is the complexity a class and } m \text{ is}$$

total number of classes.

$$ACC = (4 + 4 + 2 + 7 + 3 + 5 + 18 + 14) / 8 = 7.125 \text{ CCU}$$

i.e. the average class complexity of this project is 7.125 CCU.

Average Coupling Factor(ACF):

$$ACF = \sum_{i=1}^k CWC / k, \text{ where } CWC \text{ is the Coupling Weight for a Class}$$

and k is number of messages to other classes.

$$ACF_{GRADUATE} = 5/1 = 5$$

The average coupling factor for class Graduate is 3. There is only one method call to the outside in that class.

Average Attributes per Class(AAC):

$$AAC = \sum_{i=1}^m AC / m, \text{ where } AC \text{ is the attribute complexity and } m \text{ in the}$$

total number of classes.

$$AAC = (2 + 2 + 1 + 2 + 1 + 1 + 1 + 2)/8 = 1.5$$

i.e. the average number of attributes per class is 1.5.

5 Conclusions

A suite of object-oriented metrics are proposed in this study. The application of metric suite is shown on an example object-oriented code. These metrics are capable to capture most of the features existing in object-oriented codes such as method, attribute, class, inheritance and coupling. Further, the objective to produce such a metric suite is to combine most of the feature responsible for complexity. These metrics calculate the complexity at each level of the code and the code complexity represents the structural and cognitive complexity of an OO system.

References

1. IEEE Standard 1061-1992: Standard for a Software Quality Metrics Methodology. Institute of Electrical and Electronics Engineers, New York (1992)
2. Chidamber, S.R., Kemerer, C.F.: A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering 6, 476–493 (1994)

3. Harrison, R., Counsell, S.J., Nithi, R.V.: An Evaluation of the MOOD Set of Object Oriented Software Metrics. *IEEE Transactions on Software Engineering* 24(6), 491–496 (1998)
4. Binder, R.V.: Object-Oriented Software Testing. *Communications of the ACM* 37(9), 28–29 (1994)
5. Vaishnavi, V.K., Purao, S., Liegle, J.: Object-Oriented Product Metrics: A Generic Framework. *Information Science* 177, 587–606 (2007)
6. Purao, S., Vaishnavi, V.K.: Product Metrics for Object Oriented Systems. *ACM Computing Surveys* 35(2), 191–221 (2003)
7. Lorenz, M., Kidd, J.: *Object-Oriented Software Metrics*. Prentice Hall, Englewood Cliffs (1994)
8. Henderson-Selles, B.: *Object-Oriented Metrics, Measure of Complexity*. Prentice-Hall, Englewood Cliffs (1996)
9. Basily, V.R., Briand, L.C., Melo, W.L.: A Validation of Object Oriented Design Metrics as Quality Indicators. *IEEE Transactions on Software Engineering* 22(1), 751–761 (1996)
10. Costagliola, G., Ferrucci, F., Tortora, G., Vitiello, G.: Class Points: An Approach for the Size Estimation of Object-Oriented Systems. *IEEE Transactions on Software Engineering* 31(1), 52–74 (2005)
11. Misra, S., Akman, I.: Weighted Class Complexity: A Measure of Complexity for Object-Oriented System. *Jour. of Information Science and Engineering* 24, 1689–1708 (2008)
12. Kan, S.H.: Metrics and Lessons Learned for OO Projects, ch. 12. *Metrics and Models in Software Quality Engineering*. Addison-Wesley (2003)
13. Babsiya, J., Davis, C.G.: A Hierarchical Model for Object Oriented Design Quality Assessment. *IEEE Transactions on Software Engineering* 28(1), 4–17 (2002)
14. Briand, L., Wust, J.: Modeling Development Effort in Object Oriented System Using Design Properties. *IEEE Transactions on Software Engineering* 27(11), 963–986 (2001)
15. Kim, K., Shin, Y., Wu, C.: Complexity Measures for Object-Oriented Program Based on the Entropy. In: *Proc. Asia Pacific Software Engineering*, pp. 127–136 (1995)
16. Kim, J., Lerch, J.F.: *Cognitive Processes in Logical Design: Comparing Object-Oriented and Traditional Functional Decomposition Software Methodologies*. Carnegie Mellon University, Graduate School of Industrial Administration, Working Paper (1991)
17. Olague, H.M., Etzkorn, L.H., Gholston, S., Quattlebaum, S.: Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes. *IEEE Transactions on Software Engineering* 33(6), 402–419 (2007)
18. Pfleeger, S.L., Atlee, J.M.: *Software Engineering – Theory and Practice*. Prentice-Hall (2006)
19. Sommerville, I.: *Software Engineering*. Addison Wesley (2004)
20. Wang, Y., Shao, J.: A New Measure of Software Complexity Based On Cognitive Weights. *Canadian Journal of Electrical and Computer Engineering* 28, 69–74 (2003)

Appendix: Classes for the Case Study

```

#include <iostream.h>

/*****CLASS PERSON*****/
class person {
public:
    char * getName(){return name;}; //W_getName=1
    char * getBirthDate() {return birthDate;}; //W_getBirthDate=1
protected:
    char * name;
    char * birthDate;};

/*****CLASS EMPLOYEE*****/
class employee : public person
{
public:
    int getSalary(){return salary;}; //W_getSalary=1
    char * getSSN(){return SSN;}; //W_getSSN=1
protected:
    int salary;
    char * SSN;};

/*****CLASS STUDENT*****/
class student : public person
{
public:
    char * getMajorDept(){return majorDept;}; //W_getMajorDept=1
protected:
    char * majorDept;};

/*****CLASS STAFF*****/

class staff: public employee
{
public:
    staff(char * tname, char * tSSN, char * tbirthDate,
           int tsalary, char * tposition, bool tstudent);
    char * getPosition(){return position;}; //W_getPosition=1
    bool isStudent();
protected:
    char * position;
    bool student;};

staff::staff(char * tname, char * tSSN, char * tbirthDate,
             int tsalary, char * tposition, bool tstudent){
    name= tname; //W_staff=1
    SSN=tSSN;

```

```

    birthDate=tbirthDate;
    salary=tsalary;
    position=tposition;
    student=student;};

bool staff::isStudent(){                                     //WisStudent=1+2=3
    if (student==0)
        return false;
    else
        return true;};

/*****CLASS FACULTY *****/
class faculty: public employee
{
public:
    faculty(char * tname, char * tSSN, char * tbirthDate,
            int tsalary, char * trank);
    char * getRank(){return rank;};                       //WgetRank=1
protected:
    char * rank;};

faculty::faculty(char * tname, char * tSSN, char * tbirthDate,
                int tsalary, char * trank){               //Wfaculty=1
    name= tname;
    SSN=tSSN;
    birthDate=tbirthDate;
    salary=tsalary;
    rank=trank; };

/*****CLASS ASSISTANT *****/
class assistant: public employee
{
public:
    assistant(char * tname, char * tSSN, char * tbirthDate,
            int tsalary, short type);
    char * getType();
protected:
    short type;};

assistant::assistant(char * tname, char * tSSN,
                    char * tbirthDate, int tsalary, short ttype){
    name= tname;                                         //Wassistant=1
    SSN=tSSN;
    birthDate=tbirthDate;
    salary=tsalary;
    type=ttype; };

char * assistant::getType(){                             //WgetType=1+2=3
    if (type==1)
        return("Research assistant");
    else
        return("Teaching assistant");
};

```

```

/*****CLASS GRADUATE *****/
class graduate: public student
{
public:
    graduate(char * tname, char * tbirthDate, char * tmajorDept,
             char * tdegreeProgram);
    char * getDegreeProgram(){return degreeProgram;};
                                     //WdegreeProgram=1
    bool isStaff(staff * s);
protected:
    char * degreeProgram;};

graduate::graduate(char * tname, char * tbirthDate,
                   char * tmajorDept, char * tdegreeProgram){ //Wgraduate=1
    name=tname;
    birthDate=tbirthDate;
    majorDept=tmajorDept;
    degreeProgram=tdegreeProgram; };

bool graduate::isStaff(staff * s){ //WisStaff=1+2((2+3)+2)=15
    if (strcmp(name,s->getName())==0){
        bool result=s->isStudent();
        if (result)
            cout<<"This graduate student is an
employee"<<'\n';
        else
            cout<<"This graduate student is not an
employee"<<'\n';
        return(1);}
    else
        return(0);};

/*****CLASS UNDERGRADUATE *****/
class undergraduate: public student
{
public:
    undergraduate(char * tname, char * tbirthDate,
                  char * tmajorDept, short tclass, char * courses[6]);
    short getClass(){return sclass;}; //WgetClass=1
    short isTakenCourse(char * course);
protected:
    short sclass;
    char * courses [6];};

undergraduate::undergraduate(char * tname, char * tbirthDate,
                              char * tmajorDept, short tclass, char * tcourses[6]){
    name=tname; //Wundergraduate=1+3=4
    birthDate=tbirthDate;
    majorDept=tmajorDept;
    sclass=tclass;
    for (int i=0;i<6;i++)
        courses[i]= tcourses[i];};

```

```

short undergraduate::isTakenCourse(char * tcourse){
    for (int i=0;i<6;i++){
        //WisTakenCourse=1+3*2=7
        if (strcmp(tcourse,courses[i])==0)
            return true;
    }
    return false;};

/* =====Main Program===== */

int main ()
{
    char * courses[6];
    courses[0]="Database";
    courses[1]="OS";
    courses[2]="Programming in C";
    courses[3]="Networking";
    courses[4]="Data Structure";
    courses[5]="";
    staff * staff1 = new staff ("Aysegul Ozeke", "123456789",
        "10/05/1964", 1000, "secratery", 1);
    faculty * faculty1 = new faculty("Murat Koyuncu",
        "987654321", "10/04/1964", 4000, "Yardımcı Doçent");
    assistant * assistant1 = new assistant("Seda Camalan",
        "9876789", "10/04/1992", 1500, 2);
    graduate * graduat1 = new graduate("Aysegul Ozeke",
        "10/04/1995", "Computer", "Networking");
    undergraduate * undergraduat1 = new undergraduate("Can
        Kara", "10/04/1994", "Computer", 3, courses);
    cout<<staff1->getName()<<staff1->getSalary()<<'\n';
    cout<<faculty1->getName()<<faculty1->getSalary()<<'\n';
    cout<<assistant1->getName()<<assistant1->getType()<<'\n';
    cout<<graduat1->getName()<<graduat1->
        getDegreeProgram()<<graduat1->isStaff(staff1)<<'\n';
    cout<<undergraduat1->getName()<<undergraduat1->
        getClass()<<'\n';
    cout<<undergraduat1->isTakenCourse("Database")<<'\n';
}

```

Complexity Metrics for Cascading Style Sheets

Adewole Adewumi¹, Sanjay Misra², and Nicholas Ikhu-Omoregbe¹

¹ Department of Computer and Information Sciences, Covenant University, Nigeria

² Department of Computer Engineering, Atilim Univeristy, Ankara, Turkey
{wole.adewumi, nomoregbe}@covenantuniversity.edu.ng
smisra@futminna.edu.ng

Abstract. Web applications are becoming important for small and large companies since they are integrated with their business strategies. Cascading Style Sheets (CSS) however are an integral part of contemporary Web applications that are perceived as complex by users and this result in hampering its wide-spread adoption. The factors responsible for CSS complexity include size, variety in its rule block structures, rule block reuse, cohesion and attribute definition in rule blocks. In this paper, we have proposed relevant metric for each of the complexity factors. The proposed metrics are validated through a practical framework. The outcome shows that the proposed metrics satisfy most of the parameters required by the practical framework hence establishing them as well structured.

Keywords: Cascading Style Sheets, Complexity Metrics, Software Metrics, Validation Criteria.

1 Introduction

Web applications are becoming important for small and large companies since they are integrated with their business strategies [10]. In this point of view, it is necessary that the applications should be reliable, usable and adaptable. However, achieving this goal is not an easy task. Web applications for large scale are always being complex and therefore the maintainability of such types of system is high. There exist several quality attributes: maintainability, usability, efficiency, functionality, reliability, portability and reusability. Maintainability is one of the most important quality attribute, which must be taken care of in the development process of the web applications otherwise maintaining quality of the web application for large systems will become a challenge. One of the ways to maintain the quality is by reducing the complexity of the applications.

The complexity of software is measured in order to be able to predict the maintainability and reliability of such software. Several complexity metrics have been proposed as at today to measure typical software programs. These measures are often based on cognitive informatics [12], [17], [18], [19], [20] a way of measuring software complexity based on cognitive weights [13]. In recent times, complexity metrics have also been proposed for the web domain particularly XML schema documents

[3], Web Services [2][16] and DTDs [1]. These are an integral part of contemporary web applications. Another integral part of web applications is Cascading Style Sheets (CSS). This is a style sheet language used to format the presentation of web pages written in HTML and XHTML. In addition it can also be applied to any kind of XML document bringing about aesthetically pleasing and user-friendly interfaces. The core advantage that CSS offers is separation of content from presentation. Despite this advantage, CSS is perceived as complex by users and this result in hampering its wide-spread adoption. Though it is perceived as complex, no metric has been proposed to measure its complexity as the field of style sheets is under-researched [7]. To solve this problem in this present paper, we start in section 2 by identifying the factors that bring about complexity in a CSS document and propose relevant metrics that can be used to measure each attribute. In section 3 we demonstrate each metric using suitable example(s) and in section 4 we validate the proposed metrics through a framework. Section 5 concludes the paper.

2 Proposed Metrics

The complexity of CSS refers to how easy it is to understand and maintain. All factors that make CSS difficult to understand or maintain are responsible for its complexity. Factors that are responsible for the complexity of CSS include: size, variety in its rule block structures, rule block reuse, cohesion and attribute definition in rule blocks.

The greater the size of a CSS the more complex the CSS will be. Since size is an important measure we are proposing rule length metric which is similar to lines of code in procedural programming and number of rule block metric which is similar to the number of modules in structured programming.

Also, the more dissimilar the rule blocks in a CSS are to one another, the more complex it will be to understand. Since variety in rule block structure is an important measure, we are proposing entropy metric.

The less number of modules that are reused in CSS increases the complexity of the CSS. Since reuse is an important measure, we are proposing number of extended rule blocks metric.

Furthermore, cohesion plays a vital role in the complexity of CSS as the lower the level of cohesion among rule blocks, the more complex the CSS. Since cohesion is an important measure, we are proposing number of cohesive rule blocks as metric.

In addition, the more the attributes defined for a rule block the more complex it will be. Since attribute definition in rule blocks is an important measure, we are proposing number of attributes defined per rule block as metric. In the paragraphs that follow, we describe the proposed metrics in detail.

2.1 Rule Length (RL)

A style sheet consists of a list of rules. Rule Length (RL) metric measures the number of lines of rules (code) in a CSS. This metric does not take into account white spaces

or comment lines in the CSS. This is essentially because white spaces and comments are not executed in CSS. RL is calculated using the following formula:

$$RL = \sum \text{rule statements in a CSS file}$$

A rule statement is any of the following:

- Selector(s) + opening brace of a rule block ({} for example, **body {**
- the attribute(s) of a selector ending with a semicolon (;) for example **color: #FFFFFF;**
- Closing brace of a rule block depicted as (}

We now apply the metric to an example given in CSS code listing 1 (Figure 1).

```
/* CSS Code Listing 1 */
body {
  margin: 0;
  padding: 0;
  background: #1B120B;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 14px;
  color: #402C16;
}
h1, h2, h3 {
  margin: 0;
  padding: 0;
  font-weight: normal;
  color: #FFFFFF;
}
```

Fig. 1. CSS Code Listing 1

From CSS code listing 1, we can count 14 rule statements. Therefore,

$$RL = 14 \text{ lines}$$

2.2 Number of Rule Blocks (NORB)

A rule block refers to a selector and its attributes (properties) depicted by the syntax shown as follows.

```
/* Syntax of a rule block */
selector [, selector2, ...] [:pseudo-RLass] {
  property: value;
  [property2: value2;
  ...]
}
```

A typical CSS file will contain at least one rule block.

2.3 Entropy Metric (E)

The word ‘entropy’ was adapted from information theory [5] and is defined as a measure of uncertainty or variety. The entropy concept has been applied for the assessment of the rule complexity of procedural software [4] [6] [8] [11]. In recent times, [1] [3] have applied the concept to assess the structural complexity of XML schema documents written in W3C Document Type Definition (DTD) language and to measure the complexity of the schema documents written in W3C XML Schema Language [21] respectively. This was done by closely following the approach used by [4]. In this paper, we tow the same path to compute the entropy value of CSS documents.

According to [1] the definition is given by $E = \langle S, F, P \rangle$, where E is an experiment with S as the set of elementary events, F is a Borel field [9] over S, and P is a probabilistic function assessing real values to events in F, then for a finite number of events C_1, C_2, \dots, C_n the entropy of the given experiment E is

$$H = -\sum P(C_t) \log_2 P(C_t) \text{ where } t = 1 \dots n$$

Based on this definition the entropy of a given CSS document having n distinct class of elements can be calculated using the relative frequencies as unbiased estimates of their probabilities $P(C_i)$, $i=1, 2, \dots, n$. The distinct class of elements means that elements having the same structural complexities are grouped in the same class called equivalence class (C). This concept is demonstrated in section 3 using the CSS rule in CSS code listing 2 in Figure 2 in Appendix.

2.4 Number of Extended Rule Blocks (NERB)

This metric counts the number of rule blocks that are extended in a CSS file. It is calculated as follows:

$$NERB = \sum \text{extended rule block}(i) \text{ where } i = 1 \dots n$$

2.5 Number of Attributes Defined per Rule Block (NADRB)

This metric determines the average number of attributes defined in the rule blocks of a CSS file. It can be calculated as follows:

$$NADRB = (\text{Total no. of attributes in all rule blocks} / \text{Total no. of rule blocks})$$

2.6 Number of Cohesive Rule Blocks (NCRB)

Cohesion can be described as the “single-mindedness” of a component [10]. In the case of CSS, this can refer to rule blocks possessing a single attribute. NCRB metric counts all rule blocks that possess only one attribute. It can be calculated as follows:

$$NCRB = \sum \text{rule block } (i) \text{ possessing only one attribute where } i = 1 \dots n$$

3 Demonstration of the Proposed Metrics

For illustration, we apply all the proposed metrics from section 2 to the example given in CSS code listing 2 (Figure 2 in Appendix).

RL metric: The value of the rule length metric is 40 lines.

$$RL = 40 \text{ lines}$$

NORB metric: The value of the NORB metric is 9.

$$NORB = 9$$

Entropy metric: The entropy value of the CSS rule in CSS code listing 2 (Figure 2 in Appendix) is calculated by first determining the equivalence classes – this means grouping similar rule blocks. This is given as follows:

- C1 = {body} = 1 element
- C2 = {{h1, h2, h3}} = 1 element
- C3 = {h1, h2, h3} = 3 elements
- C4 = {a} = 1 element
- C5 = {a:hover} = 1 element
- C6 = {#page, #content} = 2 elements

The relative frequency of occurrence of the equivalence classes of the CSS document is the number of elements (i.e. attributes inside the equivalence class), divided by the total number of rule blocks in the CSS document. There are nine (9) rule blocks in the CSS document shown in CSS code listing 2 and so the relative frequency of occurrence of the equivalence class C5 = {#page, #content}, is $P(C6) = 2/9$. When all elements fall into only one equivalence class, then the minimum entropy value is determined. In that case, $P(C1) = 9/9 = 1$, this would then imply that entropy value is:

$$\begin{aligned} H &= -\sum P(C_i) \log_2 P(C_i) \\ &= P(C_1) \log_2 P(C_1) \\ &= 0 \end{aligned}$$

On the other hand, the possible maximum entropy occurs when each rule block in the CSS rule is distinct. In such a case, the number of equivalence classes equal to the number of rule blocks, i.e. $P(C_i) = 1/n$, $i = 1, 2, \dots, n$ and n is the number of rule blocks or equivalence classes in the CSS. The entropy value of the CSS rule, in this case is:

$$\begin{aligned} H &= -\sum P(C_i) \log_2 P(C_i) \\ &= -\sum (1/n) \log_2 (1/n) \end{aligned}$$

The entropy value for the CSS document shown in CSS code listing 2 is therefore calculated as:

$$\begin{aligned}
 E(\text{CSS}) &= H \\
 &= -\sum P(C_t) \log_2 P(C_t) \text{ where } t = 1..n \\
 &= (1/9) * \log_2 (1/9) + (1/9) * \log_2 (1/9) + (3/9) * \log_2 (3/9) + (1/9) * \log_2 (1/9) + \\
 &\quad (1/9) * \log_2 (1/9) + (2/9) * \log_2 (2/9) \\
 &= 0.2441 + 0.2441 + 0.3662 + 0.2441 + 0.2441 + 0.3342 \\
 &= 1.6768
 \end{aligned}$$

NERB: There are 2 extended rule blocks in CSS code listing 2 namely: h1, h2, h3 {...} and a {...}.

h1, h2, h3 {...} is extended to give h1 {...}, h2 {...} and h3 {...} while a {...} is extended to give a: hover {...}. Therefore,

$$\text{NERB} = 2$$

NADRB: There are 22 attributes in all inside CSS code listing 2. There are also 9 rule blocks in all. Applying the formula defined in section 2 we have:

$$\text{NADRB} = 22/9 = 2.44$$

In essence, the result shows that on the average two attributes are defined per rule block. The higher this value is the more complex will be the CSS.

NCRB: The total number of rule blocks that possess one attribute in CSS code listing 2 is 4 namely: h1 {...}, h2 {...}, h3 {...}, and a: hover {...}

Hence,

$$\text{NCRB} = 4$$

4 Practical Validation of the Proposed Metrics

To validate the six complexity metrics proposed we use the framework given by Kaner [14]. It is one of several validation criteria. The framework is more practical than the formal approach. It is based on answering the following points:

Purpose of the measures

The purpose of the measures is to evaluate the complexity of cascading style sheets.

Scope of usage of the measure: The proposed RL, NORB, entropy, NERB, NADRB and NCRB metrics are good predictors of understandability of CSS. They are

therefore a valuable contribution for maintainability of CSS. The scope of use is by web development teams that work on styling web interfaces.

Identified attribute to measure

The identified attributes to measure from our suite of metrics are understandability, reliability and maintainability. All these attributes are directly related to the quality of CSS.

Natural scale of the attribute

The natural scales of the attributes cannot be defined, since they are subjective and require the development of a common view about them.

Natural variability of the attribute

Natural variability of the attributes can also not be defined because of their subjective nature. It is possible that one can develop a sound approach to handle such attribute, but it may not be complete because other factors also exist that can affect the attribute's variability. In this respect, it is difficult to attain knowledge about variability of the attribute.

Definition of metric

The metrics have been formally defined in Section 2.

Measuring instrument to perform the measurement: We have counted all the parameters of the metrics manually and computed the proposed metrics. Further, we aim at developing a tool/software for measuring the proposed suite of metrics.

Natural scale for the metrics

For the natural scale of the proposed metrics, we have to go through measurement theory. When we analyze our metrics according to Briand and Morasca [15] we find that, they are in the ratio scale.

The natural variability of readings from the instrument

Since the reading from our counting instrument is not subjective and does not require any interpretation, we can say that no variability (i.e. measurement error) on readings from the instrument can be expected. Note that, in case of automated counting, we assume that there is no bug in the devised algorithm.

Relationship between the attribute to the metric value

There is a direct relation between the complexity of CSS and our proposed metrics. In other words all the proposed metrics are predictors of complexity in CSS.

Natural and foreseeable side effects of using the instrument

Once we automate the complexity calculation, it will not require considerable additional workload of manpower of the company. The only cost will be the automation.

Table 1. Results of applying proposed metrics to CSS code listing 2

Metrics	Code Listing 2
RL	40
NORB	9
Entropy	1.6768
NERB	2
NADRB	2.44
NCRB	4

5 Concluding Remark and Further Work

In this paper, we identified factors that bring about complexity in CSS and also proposed complexity metrics based on each of these factors for analyzing the complexity of CSS documents. With these proposed metrics, Web developers and designers can measure the complexity of CSS documents in terms of size, variety in rule block structure, rule block reuse, cohesion and the average number of attributes defined per rule block. The proposed metrics were validated practically through a framework to prove their usefulness and practical applicability. It was found that the proposed metric satisfies most of the parameters required by the practical evaluation framework.

As future work, we intend to validate each metric through Weyuker's properties. Rigorous empirical validation will also be done. In addition, the development of an automated tool for computing the metrics is also a task of future work.

References

1. Basci, D., Misra, S.: Entropy Metric for XML DTD Documents. *ACM SIGSOFT Software Engineering Notes* 33(4) (2008)
2. Basci, D., Misra, S.: Data Complexity Metrics for XML Web Services. *Advances in Electrical and Computer Engineering* 9(2) (2009)
3. Basci, D., Misra, S.: Entropy as a Measure of Quality of XML Schema Document. *The International Arab Journal of Information Technology* 8(1), 16–24 (2011)
4. Davis, J., LeBlanc, R.: A study of the applicability of complexity measures. *IEEE Transaction on Software Engineering* 14, 366–372 (1988)
5. Hamming, R.: *Coding and information theory*. Prentice Hall, Englewood Cliffs (1980)
6. Harrison, W.: An entropy-based measure of software complexity. *IEEE Transactions on Software Engineering* 18, 1025–1029 (1992)
7. Marden, P.M., Munson, E.V.: *Today's Style Sheet Standards: The Great Vision Blinded*. Computer (1999)
8. Mohanty, S.N.: Entropy metrics for software design evaluation. *The Journal of Systems and Software* 2, 39–46 (1981)
9. Papoulis, A.: *Probability, random variables and stochastic processes*. McGraw-Hill, New York (1965)
10. Pressman, R.S.: *Software Engineering: A Practitioner's Approach*. McGraw-Hill, New York (2005)

11. Torres, W., Samadzadeh, M.H.: Software reuse and information theory based metrics. *IEEE Transactions on Software Engineering*, 437–446 (1990)
12. Wang, Y.: On Cognitive Informatics. In: *Second IEEE International Conference on Cognitive Informatics (ICCI 2002)*, pp. 34–42 (2002)
13. Wang, Y, Shao, J.: A New Measure of Software Complexity based on Cognitive Weights. *Can. J. Electrical and Computer Engineering*, 69–74 (2003)
14. Kaner, C.: Software Engineering Metrics: what do they measure and how do we know? In: *Proc. Tenth Int. Software Metrics Symp., Metrics*, pp. 1–10 (2004)
15. Briand, L.C., Morasca, S., Basily, V.R.: Property based software engineering measurement. *IEEE Transactions on Software Engineering* 22, 68–86 (1996)
16. Basci, D., Misra, S.: Metrics Suite for Maintainability of XML Web-Services. *IET Software* 5(3), 320–341 (2011)
17. Misra, S., Cafer, F.: Estimating Complexity Of Programs In Python Language. *Technical Gazette* 18(1), 23–32 (2011)
18. Misra, S., Akman, I., Koyuncu, M.: An Inheritance Complexity Metric for Object Oriented Code: A Cognitive Approach. *SADHANA* 36(3), 317–338 (2011)
19. Misra, S., Akman, I.: Unified Complexity Measure: a measure of Complexity. *The Proc. National Academy of Sciences India (Sect. A)* 80(2), 167–176 (2010)
20. Misra, S., Akman, I.: Weighted Class Complexity: A Measure of Complexity for Object Oriented Systems. *Journal of Information Science and Engineering* 24, 1689–1708 (2008)
21. Basci, D., Misra, S.: Measuring and Evaluating a Design Complexity Metric for XML Schema Documents. *Journal of Information Science and Engineering* 25(5), 1405–1425 (2009)

Appendix: 1

```

/* CSS Code Listing 2 */
body {
    margin: 0;
    padding: 0;
    background: #FFFFFF;
    font-family: Arial, Helvetica, sans-serif;
    font-size: 14px;
    color: #402C16;
}
h1, h2, h3 {
    margin: 0;
    padding: 0;
    font-weight: normal;
    color: #2E9F13;
}
h1 {
    font-size: 2em;
}
h2 {

```



```
    font-size: 2.4em;
}
h3 {
    font-size: 1.6em;
}
a {
    text-decoration: none;
    color: #2E9F13;
}
a:hover {
    text-decoration: underline;
}
/* Page */
#page {
    width: 960px;
    padding: 0;
    border-top: 1px solid #D0D0D0;
}
/* Content */
#content {
    float: right;
    width: 600px;
    padding: 0px 0px 0px 0px;
}
```

Fig. 2. CSS Code Listing 2

A Systematic Review on the Impact of CK Metrics on the Functional Correctness of Object-Oriented Classes

Yasser A. Khan, Mahmoud O. Elish, and Mohamed El-Attar

Department of Information & Computer Science
King Fahd University of Petroleum & Minerals
Dhahran, Saudi Arabia
{yasera, elish, melattar}@kfupm.edu.sa

Abstract. The Chidamber and Kemerer (CK) metrics suite is one of the most popular and highly cited suites for measuring Object-Oriented (OO) designs. A great amount of empirical studies have been conducted to evaluate these metrics as indicators of the functional correctness of classes in OO systems. However, there has been no attempt to systematically review and report these empirical evidences. To identify the relation of CK metrics with functional correctness, we have performed a systematic review of empirical evidences published in the literature that support or reject CK metrics as indicators of functional correctness. Our search strategy identified 20 papers that contain relevant empirical evidences. Our results conclude that WMC, CBO, RFC and LCOM metrics are good indicators of functional correctness of OO classes. Inheritance metrics, DIT and NOC, are however not useful indicators of functional correctness.

Keywords: Software quality, CK metrics, functional correctness, systematic literature review.

1 Introduction

The Chidamber and Kemerer (CK) Object-Oriented (OO) design metrics suite was introduced in 1994 as an alternative to traditional, non OO, software metrics [1]. The traditional metrics did not capture OO features such as inheritance, coupling and cohesion. The CK metrics suite was introduced especially to measure these features. It is strongly rooted in theory, and applicable to OO development practices. Popular software development tools such as Rational Rose have incorporated the CK metrics for OO design measurement.

The ISO/IEC 25010 [10] standard defines eight characteristics of software product quality—functional suitability, maintainability, performance efficiency, compatibility, usability, security, reliability and portability. These characteristics are further divided into sub-characteristics. *Functional correctness* is a sub-characteristic of the functional suitability characteristic. It is defined as “the degree to which a product or system provides the correct results with the needed degree of precision” [9]. Inverted proxy measures for functional correctness include fault count, fault proneness, and fault density and severity.

Several studies have empirically validated the CK metrics as indicators of the functional correctness of OO classes. To the best of our knowledge, there has been no effort to systematically report the empirical evidences in literature that support or reject the CK metrics as indicators of functional correctness. Therefore, there is a need to conduct a Systematic Literature Review (SLR) of the empirical evidences in literature that are in favor or against the CK metrics as indicators of the functional correctness of OO classes. This paper reports the methodological details and the results of our SLR.

The remainder of this paper is organized as follows. Section 2 contains an introduction to the CK metrics suite. Section 3 reports the details of our SLR research methodology. Section 4 contains an overview of the selected primary studies. Main findings of our SLR are presented in Section 5. Section 6 discusses the limitations of the review and Section 7 concludes the paper.

2 CK Metrics Suite

The CK metrics suite [1] is one of the most popular and highly cited suites for measuring OO designs. These metrics help designers in evaluating their design and detecting design flaws. The suite consists of the following six metrics:

1. *Weighted methods per class (WMC)*: The sum of individual complexity of each method in a given class.
2. *Depth of inheritance tree (DIT)*: The maximum length from the root to a given class in an inheritance hierarchy.
3. *Number of children (NOC)*: The number of direct subclasses of a given class in an inheritance hierarchy.
4. *Coupling between object classes (CBO)*: The number of classes that a class is coupled with.
5. *Response for a class (RFC)*: The number of methods in a class plus number of distinct methods called by those methods.
6. *Lack of Cohesion in methods (LCOM)*: The number of methods pairs in a class that share no instance variables minus number of methods pairs that share instance variables.

3 Research Method

The research has been carried out by following the guidelines given by Kitchenham and Charters [2] for conducting SLR in Software Engineering. SLR is a well-defined methodical way of identifying, assessing and analyzing relevant published literature in order to answer a research question. The remainder of Section 3 discusses our SLR procedure in detail.

3.1 Research Questions

Several empirical studies have validated the effectiveness of the CK metrics as indicators of functional correctness of OO classes. Our goal is to provide an aggregate of the empirical evidences reported in literature that support/reject the CK metrics as indicators of functional correctness. Therefore, our first research question is:

RQ1 - Are CK metrics significant indicators of the functional correctness of OO classes?

If a CK metric appears as an indicator of the functional correctness of OO classes, it would be interesting to identify the nature of the relationship—positive or negative. A positive relationship implies higher the measure of the metric higher the class quality in terms of functional correctness. A negative relationship implies higher the measure of the metric lower the class quality in terms of functional correctness and vice versa. Therefore, our second research question is:

RQ1.1 - What is the nature of the relationships between an individual CK metric and the functional correctness of OO classes?

3.2 Search Strategy

A search strategy aims to make sure that all relevant literature appears in the search result without cluttering up with irrelevant studies. We limited our literature search over two dimensions: publication time period and publications that have cited CK's original paper [1]. In terms of publication time period we limited our search to the papers published between June 1994 and October 2011. This start date was chosen because the CK metrics suite paper [1] was published in June 1994. The end date is 30 October 2011 since we finished our search on this date. Therefore, any paper published after 30 October 2011 is not included in our search result. We limited our search to the electronic database Scopus as it covers the most relevant and respected journals and conferences proceedings within software engineering. It contained 1223 papers that cited the CK metrics suite paper [1]. Initially, papers were excluded based on relevance of their title and abstract to our research questions. The remaining papers were then selected based on our inclusion criteria—studies that empirically investigate the relationship between CK metrics or a subset of CK metrics with functional correctness. Based on this, 20 papers were selected as the primary studies for this SLR. They are listed in the Appendix. It worth mentioning that at least two members of the research team were involved independently in the study selection process. Any discrepancies were settled by consensus.

3.3 Data Extraction and Synthesis

The selected studies were read in depth by at least two researchers in parallel in order to extract the data needed to address the research questions. Important lines and

paragraphs in the selected studies were highlighted. This helped us to quickly find and validate the extracted information and resolve inconsistencies. Data was extracted in a detailed data extraction form. The fields of our data extraction form are study title, journal/conference, date of publication, author name(s), study objective, CK metric(s) measured, proxy metric(s) used to measure functional correctness, system information, data analysis methodology, metrics measurement method and results of the study.

In data synthesis step, the extracted data from the extraction forms is compiled and organized in a suitable form to answer the research questions. We performed descriptive synthesis of the data and presented it in tabular form for each of the CK metrics (Tables 2-7).

4 Overview of Selected Studies

The 20 selected studies based on the search strategy and inclusion criteria investigate the relationship between the CK metrics or a subset of CK metrics and functional correctness. Functional correctness can be quantified through inverted proxy metrics such as fault count, fault proneness, fault density, etc. Higher the faultiness of a class the lower will be its quality in terms of functional correctness. [S13], [S18] and [S19] are few of the studies that investigated the relationship between the CK metrics and class-level Fault Count (FC). Fault Proneness (FP) refers to probability of occurrence of a fault in a class. [S1], [S7] and [S20] are few of the studies that investigated the relationship between the CK metrics and FP. The relationship between Fault Severity (FS) (high, medium, low and ungraded) and the CK metrics was investigated in [S5]. Fault Density (FD) refers to the fault count per lines of code (LOC). Its relationship with the CK metrics was investigated in [S19]. Vulnerability Count (VC) was correlated with the CK metrics in [S2]. A class is classified as vulnerable if it contains faults that violate a system's security policy. Table 1 summarizes the proxy metrics used by the selected studies.

The different types of systems investigated by the selected studies are applications, components and middleware. They include both industrial and open-source systems and, systems developed by students. They are of variable size ranging from small to large sized categories. C++ is the most frequently used programming language in the analyzed systems. Sixteen of the analyzed systems are written in C++. Thirteen of them are written in Java. It is interesting to note that none of the studies analyzed systems written in other popular OO languages such as C#, Visual Basic, Smalltalk etc. The different analysis methodologies used by the studies to find a relationship between CK metrics and functional correctness are as follows. FP and FS are predicted using logistic regression. Linear regression and correlation analysis are used for investigating the effect of CK metrics on FC, FD and VC.

Table 1. Proxy metrics used by the selected studies

Proxy	Definition	Study
Fault Count (FC)	Count of faults in a class.	[S4], [S6], [S7], [S8], [S10], [S11], [S12], [S13], [S14], [S18], [S19]
Fault Proneness (FP)	Probability of occurrence of a fault in a class.	[S1], [S3], [S4], [S7], [S8], [S9], [S10], [S11], [S12], [S14], [S15], [S16], [S17], [S20]
Fault Severity (FS)	Fault Proneness categorized as high (HSF), medium (MSF), low (LSF) and ungraded (USF) severity	[S5]
Fault Density (FD)	Fault count per LOC.	[S19]
Vulnerability Count (VC)	Count of faults that can violate a system's security policy	[S2]

5 Results and Discussion

This section discusses the empirical evidences reported in the selected studies, and answers the research questions for all six of the CK metrics. Tables 2-7 summarize results of the selected studies for each of the six CK metrics. They list the proxy metrics and references to the studies that have shown positive, negative and insignificant effect of a metric on functional correctness. Studies which have shown contradictory results for different systems are represented more than once in the same row.

5.1 WMC

WMC is a measure of class complexity. It is given as the weighted sum of individual complexities of local methods in a class. In the remainder of this section we refer to WMC as the version of this metric in which method complexity for all methods is taken as unity. The version which computes this metric as the sum of McCabe's complexity factor [4] of each method is referred as WMC-McCabe. Table 2 summarizes the results of the selected studies with respect to WMC.

Impact on FC. WMC showed moderate to large correlation coefficients with FC for four versions of Mozilla Rhino, 14R3, 15R1, 15R3 and 15R4 in [S8]. However, small correlation coefficients were reported for two versions, 15R2 and 15R5. The correlation coefficients were classified based on the guidelines by Cohen [3]. WMC-McCabe showed moderate to large coefficients for five versions except 15R5. However, in [S11], small to large correlation coefficients were reported for versions 14R3, 15R1 and 15R2 according to the Hopkins scale [8]. Harrison et al. [S19]

reported significant correlations with number of errors found during system and integration testing for two student projects. However, four similar student projects showed insignificant correlations. The average class size of these student projects is, 12, very low so we cannot generalize these results to real world software systems. They also reported insignificant correlation for an image analysis subsystem. Subramanyam and Krishnan [S13] analyzed a commercial OO system which was composed of two sub-systems written in different languages. The C++ subsystem showed a positive relationship whereas the Java subsystem showed insignificant relationship. [S13] also reported support for varied results across programming languages. The Java subsystem contained lower average LOC per class compared to the C++ subsystem. Additionally, the Java subsystem showed stronger correlation with class size (LOC) compared to the C++ subsystem. The weak correlations in [S8], [S11] and [S19] may be clarified by investigating the effect of class size on WMC. [S6], [S7] and [S10] analyzed the same metrics dataset and reported consistent support for WMC as an indicator of FC. Open source systems, in [S4] and [S12], also showed likewise results. Kamiya et al. [S18] investigated an inventory control system and reported a significant correlation. Hence, it can be concluded that WMC is a good indicator of FC. A class with WMC higher than its peers is at risk of containing more faults as compared to them. However, as shown in [S13], the effect of WMC on class size must be investigated prior to establishing a relationship with FC.

Impact on FP. WMC and WMC-McCabe were significant predictors of faulty classes for six versions of Mozilla Rhino [S8]. These results were reported again in [S11]. Basili and Briand [S20] analyzed eight student projects as a whole and showed that WMC is a significant indicator of FP. [S7] and [S10] analyzed the same dataset and showed similar effect of WMC on FP. El Emam et al. [S15] also showed similar results but the effect of WMC on FP diminished after controlling for class size, thus indicating a confounding effect of class size on the relationship between WMC and FP. WMC was the best predictor of faulty classes in [S12] for Mozilla version 1.6. WMC was a reasonably good predictor of faulty classes for the Java Development Kit (JDK) [S4]. The evidence in [S9] for student projects showed similar results. Hence, it can be concluded that WMC is a good predictor of faulty classes. A class with WMC higher than its peers is more fault-prone as compared to them. However, as mentioned in [S15] the confounding effect of class size on the association between WMC and FP must be investigated prior to establishing a relationship between WMC and FP.

Impact on FS, FD, and VC. WMC was significant for predicting HSF, MSF, LSF and USF [S5]. Harrison et al. [S19] reported an insignificant correlation with FD for an image analysis subsystem. They also reported insignificant correlations for five student projects. However, WMC was significant for one similar student project. As discussed earlier, these results cannot be generalized to real world software systems. Five releases of Mozilla Firefox showed moderate correlations between WMC and VC [S2].

From the above reported empirical evidences it can be concluded that WMC has a negative impact on the functional correctness of a given class. However, the impact of WMC on FD cannot be concluded.

Table 2. WMC and functional correctness – Results summary of selected studies

Proxy	Positive	Negative	Insignificant
FC	-	[S4], [S6], [S7], [S8], [S10], [S11], [S12], [S13] [S18], [S19]	[S8], [S13], [S19]
FP	-	[S4], [S7], [S8], [S9], [S10], [S11], [S12], [S15], [S20]	-
HSF, MSF, LSF, USF	-	[S5]	-
FD	-	[S19]	[S19]
VC	-	[S2]	-

5.2 DIT

DIT is the depth of a class in an inheritance hierarchy. This section answers the research questions with respect to DIT. Table 3 summarizes the results of the selected studies with respect to DIT.

Impact on FC. DIT was not a useful indicator of FC in [S4]. Correlation and regression analysis both revealed it as a not useful indicator in [S6]. [S10] analyzed the same dataset, as in [S6], and supported DIT as an insignificant indicator of FC. Olague et al. [S11] reported insignificant correlation for two version of Mozilla Rhino–15R3 and 15R5. However, they reported a moderate correlation coefficient for one version, 15R4. Five student projects showed insignificant correlation coefficients in [S19]. However, one of them was negatively moderate. As mentioned previously, these results cannot be generalized to real world systems. Two commercial systems showed insignificant correlation coefficients between DIT and FC in [S14]. On the contrary, Kamiya et al. reported large correlation with FC in [S18] for a C++ system. Another C++ system showed high DIT values were associated with higher FC [S13]. The results for these two systems can be explained by the fact that C++ supports multiple inheritance. A class which inherits attributes and methods from multiple classes will be more complex than classes which inherit from a single class. However, datasets taken from other C++ systems showed DIT was an insignificant indicator of FC [S6] [S10] [S14]. Multiple inheritance may not have been used as much in these systems as used in [S13] and [S18]. Hence, we conclude that DIT is not related with FC. However, the impact of multiple inheritance on FC requires further investigation.

Impact on FP. DIT was shown not related to FP in [S7], [S9] and [S15]. Four releases of Mozilla Rhino also showed insignificant effect of DIT on FP [S11]. A dataset consisting of student projects showed high DIT associated with high FP [S17]. Inheritance was sparingly used in these datasets; hence, these results cannot be generalized to real world systems. Gyimothy et al. [S12] reported DIT was significant

for fault prediction. However, they reported that DIT's accuracy in predicting faults was very poor compared to other CK metrics. The evidence in [S4] also showed DIT as poor predictor of faults compared to other CK metrics. [S16] and [S20] concluded DIT was very significant, however its accuracy in predicting faults was not tested. Hence, we conclude that DIT is not useful for predicting faulty classes.

Impact on FS, FD, and VC. DIT was insignificant for predicting HSF, MSF, LSF and USF in [S5]. It showed insignificant correlation coefficients with FD for five student projects [S19]. As discussed earlier, these results cannot be generalized to real world software systems. Five releases of Mozilla Firefox showed moderate correlations between DIT and VC [S2].

From the above reported empirical evidences it can be concluded that DIT has no impact on the functional correctness of a given class.

Table 3. DIT and functional correctness – Results summary of selected studies

Proxy	Positive	Negative	Insignificant
FC	-	[S13], [S11], [S18]	[S4], [S6], [S10], [S11] [S14], [S19]
FP	-	[S4], [S11], [S12], [S16], [S17], [S20]	[S7], [S9], [S11], [S15]
HSF, MSF, LSF, USF	-	-	[S5]
FD	-	-	[S19]
VC	-	[S2]	-

5.3 NOC

NOC is the number of direct subclasses of a given class in an inheritance hierarchy. This section answers the research questions with respect to NOC. Table 4 summarizes the results of the selected studies with respect to NOC.

Impact on FC. NOC gave small to moderate correlation coefficients with FC for three versions of Mozilla Rhino—15R3, 15R4 and 15R5 [S11]. One student project in [S19] showed insignificant correlation coefficients. Correlation and regression analysis both revealed it as a not useful indicator of FC in [S6]. [S10] analyzed the same dataset, as in [S6], and concluded similar. Two commercial systems showed insignificant correlations between NOC and FC [S14]. NOC was not a good indicator of FC for JDK in [S4]. Hence, we conclude that NOC is not related with FC.

Impact on FP. NOC was associated with lower probability of fault detection for a dataset comprising of student projects [S20]. The dataset showed a flat inheritance structure, most of the classes had no children; hence this result cannot be generalized to real world systems. Another student project dataset showed similar results [S17]. On the contrary, in [S9] a student project dataset showed insignificant effect of NOC on FP [S9]. NOC was least significant for predicting faulty classes compared to other

metrics in [S4], [S7], [S12] and [S16] showed NOC was not useful for predicting faulty classes. Four version of Mozilla Rhino showed insignificant effect of NOC on fault-proneness [S11]. Hence, we conclude that NOC is not related to FP.

Impact on FS, FD, and VC. NOC was not useful for predicting HSF and LSF whereas it was associated with lower probability of MSF and USF [S5]. One student project in [S19] showed an insignificant correlation between NOC and FD. As discussed earlier, this result cannot be generalized to real world software systems. Five releases of Mozilla Firefox showed strong correlations between NOC and VC [S2].

From the above reported empirical evidences it can be concluded that NOC has no impact on the functional correctness of a given class.

Table 4. NOC and functional correctness – Results summary of selected studies

Proxy	Positive	Negative	Insignificant
FC	-	[S11]	[S4], [S6], [S10], [S14], [S19]
FP	[S17], [S20]	[S4], [S11]	[S9], [S7], [S11], [S12], [S16]
HSF, LSF	-	-	[S5]
MSF, USF	[S5]	-	-
FD	-	-	[S19]
VC	-	[S2]	-

5.4 CBO

Two classes are said to be coupled if one of them accesses an instance variable or calls a method from the other. CBO is the count of the number of classes a given class is coupled with. It gives a measure of interdependence between OO system classes. This section answers the research questions with respect to CBO. Table 5 summarizes the results of the selected studies with respect to CBO.

Impact on FC. Harrison et al. [S19] reported an insignificant correlation between CBO and FC for four student projects. However, two similar projects showed significant correlations. As discussed earlier, these results cannot be generalized to real world software systems. They also reported an insignificant correlation for an image analysis subsystem. Succi et al. [S14] reported conflicting results for two commercial C++ systems—Project A and Project B; the former showed insignificant correlation whereas the latter showed significant correlation. These contradicting results were explained by the fact that less than 30% of classes in Project A correspond to 80% of faults, whereas only 2% of classes correspond to 80% of faults in Project B. Subramanyam and Krishnan [S13] also showed contradictory results for two components of a commercial system. The C++ subsystem showed a positive relationship whereas the Java subsystem showed a negative relationship. This was reported due to the fact that the Java subsystem showed an interaction between DIT and CBO. As the depth of a class in an inheritance hierarchy increased, the effect of CBO on fault count decreased. However, root level classes (DIT=0) showed an

increase in faults with the increase in CBO. The insignificant correlation reported in [S19] for an image analysis subsystem may be clarified by investigating the interaction between DIT and CBO. The design of this system made no use of inheritance (DIT=0 for all classes). CBO was a good indicator of FC for the JDK in [S4]. Kamiya et al. [S18] analyzed an inventory control system and reported a significant correlation. However, CBOR (coupling with framework classes), showed a stronger correlation coefficient than CBO. This means that coupling with framework classes contributes more faults compared to coupling with newly developed classes. Olague et al. [S11] reported small to moderate correlation coefficients for three versions of Mozilla Rhino, 15R3, 15R4, and 15R4, according to the Hopkins scale [8]. Moreover, the correlation coefficients were larger compared to direct class coupling (DCC) of the QMOOD metrics suite [7]. [S6], [S7] and [S10] analyzed the same dataset and showed consistent support for CBO as an indicator of FC. Open source web and email suite, Mozilla version 1.6, also showed likewise results in [S12]. Hence, it can be concluded that CBO is a good indicator of FC. A class which is more coupled than its peers is at risk of containing more faults as compared to them. However, as shown in [S13] the interaction between DIT and CBO must be investigated prior to establishing a relationship between CBO and FC.

Impact on FP. Olague et al. [S11] reported CBO as a significant indicator of FP for five out of six versions of Mozilla Rhino. Moreover, it performed as a better indicator than DCC [7]. [S9], [S17] and [S20] analyzed student projects and supported CBO as a significant indicator. El Emam et al. [S15] also showed similar results but the effect of CBO on FP diminished after controlling for class size, thus indicating a confounding effect of class size on the relationship between CBO and FP. [S7] reported CBO as a better predictor of FP than fault count. [S10] analyzed the same dataset, as in [S7], and concluded that CBO was significant for predicting faulty classes. The correctness, precision and completeness of CBO in predicting faults were highest compared to other metrics in [S4]. Precision and completeness in predicting faults were also highest in [S12]. Hence, it can be concluded that CBO is a good indicator of FP. A class which is more coupled than its peers is more fault-prone as compared to them. However, as mentioned in [S15] the confounding effect of class size on the association between CBO and FP must be investigated prior to establishing a relationship between CBO and FP.

Impact on FS, FD, and VC. CBO was significant for predicting HSF, MSF, LSF and USF in [S5]. Harrison et al. [S19] reported an insignificant correlation with FD for an image analysis subsystem. They also reported positive correlation coefficients for four student projects. On the contrary, two similar projects showed negative correlation coefficients. As discussed earlier, these results cannot be generalized to real world software systems. CBO showed moderate correlation with VC in [S2].

From the above reported empirical evidences it can be concluded that CBO has a negative impact on the functional correctness of a given class. However, the impact of CBO on FD cannot be concluded.

Table 5. CBO and functional correctness – Results summary of selected studies

Proxy	Positive	Negative	Insignificant
FC	[S13]	[S4], [S6],[S7], [S10], [S11], [S12], [S13], [S14],[S18], [S19]	[S14], [S19]
FP	-	[S4], [S7], [S9], [S10], [S11], [S12], [S15], [S17], [S20]	[S11]
HSF, MSF, LSF, USF	-	[S5]	-
FD	[S19]	[S19]	[S19]
VC	-	[S2]	-

5.5 RFC

RFC is size of the response set of a class. Response set includes class methods and set of distinct methods that are called by the class methods. This section answers the research questions with respect to RFC. Table 6 summarizes the results of the selected studies with respect to RFC.

Impact on FC. RFC was the best predictor of FC compared to other CK metrics in [S4]. Succi et al. [S14] reported conflicting results for two commercial C++ systems—Project A and Project B; the former showed insignificant correlation whereas the latter showed significant correlation. These contradicting results were explained by the fact that less than 30% of classes in Project A correspond to 80% of faults, whereas only 2% of classes correspond to 80% of faults in Project B. RFC gave strong correlation coefficient with FC in [S18]. However, a variant RFCN, that considers methods from newly developed classes and excludes methods from framework classes, gave stronger correlation. This shows that reuse of newly developed classes can contribute more faults than reuse of framework classes. [S6], [S7] and [S10] analyzed the same dataset and supported RFC as good indicator of FC. Three versions of Mozilla Rhino showed significant correlation coefficients between RFC and FC [S11]. Hence, we conclude that RFC is a good indicator of FC. A class with RFC higher than its peers is at risk of containing more faults as compared to them.

Impact on FP. RFC was very significant for predicting faults in [S20]. The correctness of RFC in predicting faults was high compared to other metrics in [S4]. Its precision in predicting faults was higher than other CK metrics (except CBO) in [S12]. RFC was reported as a better indicator of FP than FC in [S7]. [S10] analyzed the same dataset, as in [S7], and concluded that RFC was significant for predicting faulty classes. El Emam et al. [S15] also showed similar results but the effect of RFC on FP diminished after controlling for class size, thus indicating a confounding effect of class size on the relationship between RFC and FP. RFC was a significant predictor of faults for six versions of Mozilla Rhino [S11]. Student projects analyzed in [S9] and [S17] also concluded significant effect of RFC in predicting faults. Hence, we conclude that RFC is a good predictor of FP. A class with RFC higher than its peers is

more fault-prone as compared to them. However, as mentioned in [S15] the confounding effect of class size on the association between RFC and FP must be investigated prior to establishing a relationship between RFC and FP.

Impact on FS and VC. RFC was significant for predicting HSF, MSF, LSF and USF in [S5]. Five releases of Mozilla Firefox showed moderate correlations between RFC and VC [S2].

From the above reported empirical evidences it can be concluded that RFC has a negative impact on the functional correctness of a given class.

Table 6. RFC and functional correctness – Results summary of selected studies

Proxy	Positive	Negative	Insignificant
FC	-	[S4], [S6], [S7], [S10], [S11], [S14], [S18]	[S14]
FP	-	[S4], [S7], [S9], [S10], [S11], [S12], [S15],[S17], [S20]	-
HSF, MSF, LSF, USF	-	[S5]	-
VC	-	[S2]	-

5.6 LCOM

Cohesion refers to the degree of interrelatedness among the components of a class—instance variables and methods. LCOM is an inverted measure of class cohesion. A high LCOM indicates a less cohesive class and vice versa. This section answers the research questions with respect to LCOM. Table 7 summarizes the results of the selected studies with respect to LCOM.

Impact on FC. Succi et al. [S14] reported conflicting results for two commercial C++ systems—Project A and Project B. The former showed insignificant correlation between LCOM and FC whereas latter showed a significant correlation. These contradicting results were explained by the fact that less than 30% of classes in Project A correspond to 80% of faults, whereas only 2% of classes correspond to 80% of faults in Project B. Three versions of Mozilla Rhino showed small to moderate correlation coefficients between LCOM and FC [S11]. Kamiya et al. [S18] analyzed an inventory management system and reported significant correlation between LCOM and FC. Hence, it can be concluded that LCOM is a good indicator of FC. A class having lower cohesion than its peers is at risk of containing more faults as compared to them.

Impact on FP. Briand et al. [S17] reported LCOM as the least significant indicator of faulty classes compared to its variants and other cohesion measures. The Hitz and Montazeri [5] and the Henderson-Sellers [6] variants of LCOM were significant indicators. These variants of LCOM performed better because LCOM does not differentiate between cohesive classes (LCOM=0). Basili et al. [S20] also reported

LCOM as an insignificant indicator due to this limitation. LCOM was again insignificant for predicting faults in [S15]. On the contrary, Gyimothy et al. [S12] analyzed open source systems and reported LCOM as a good predictor of faulty classes. LCOM's effectiveness against its negative version (LCOMN) was also investigated but reported results could not conclude which one was better. Moreover, their correctness in predicting faults were highest compared to other CK metrics. Al-Dallal [S1] proposed TLCOM (transitive LCOM) that considers indirect relations between class attributes and methods and investigated its effectiveness in predicting faults. The results revealed it as a better indicator of FP than LCOM. Al-Dallal and Briand [S3] proposed a similarity based cohesion metric and empirically validated it as a better fault predictor than LCOM. Aggarwal et al. [S9] analyzed student projects and supported LCOM as a good indicator. It could be argued that these results cannot be generalized to real world systems. Olague et al [S11] reported LCOM as a significant indicator of FP for four versions of Mozilla Rhino. However, two versions showed insignificant relationship. Studies have shown that variants of LCOM are better indicators of FP ([S1], [S3] and [S17]). Due to the contradictory results and limitations of LCOM in predicting faulty classes we can conclude that LCOM is not an indicator of FP.

Impact on FD. Harrison et al. [S19] reported LCOM as an insignificant indicator of FD for an image analysis subsystem.

From the above reported empirical evidences we conclude that LCOM has a negative impact on the functional correctness of a given class when FC is used as a proxy. However, the impact of LCOM on FP and FD cannot be determined.

Table 7. LCOM and functional correctness – Results summary of selected studies

Proxy	Positive	Negative	Insignificant
FC	-	[S11], [S14], [S18]	[S14]
FP	-	[S1], [S3], [S9], [S11], [S12]	[S11], [S15], [S17], [S20]
FD	-	-	[S19]

6 Limitations

This SLR has some limitations which should be considered while interpreting its results. Since we restricted our literature search to Scopus this SLR may have not included every published study that can answer the research questions. Moreover, our search string may have missed some relevant studies. We also found that some papers which performed correlation analysis lacked sufficient data to support their results — p-values, significance threshold and level of significance. Some papers did not mention whether the correlation coefficients were statistically significant or not. For such papers, we used the guidelines given in [3] to classify the coefficients as small, moderate or large. Descriptive statistics of the systems investigated were also missing in few papers.

7 Conclusion

Table 8 summarizes the results of this SLR by indicating the relationships between the CK metrics and the functional correctness of OO classes through the inverted proxies. It should be noted that relationships are given as positive (+ve) or negative (-ve) with respect to functional correctness. A positive relationship implies higher the measure of the metric higher the class quality in terms of functional correctness. A negative relationship implies higher the measure of the metric lower the class quality in terms of functional correctness and vice versa. 'nr' indicates that the metric is not useful for predicting functional correctness through the corresponding proxy metric. Relationships that require further empirical investigation are given by 'nc'. An empty cell indicates that none of the selected studies investigated the relationship between the CK metric and the proxy metric.

As a result of this SLR we conclude that certain CK metrics are indeed good indicators of functional correctness. WMC, CBO, RFC and LCOM have a negative relationship with the functional correctness of a class. However, LCOM is useful for predicting FC rather than classifying classes as faulty. Inheritance metrics, DIT and NOC, are not related to the functional correctness of a class. The relationship between CK metrics and FD also could not be determined.

This SLR can help software engineering practitioners in making informed decisions about the impact of CK design metrics on the functional correctness of OO classes. For researches, this SLR provides an aggregation of the reported empirical evidences on the usefulness of the CK metrics as indicators of software quality in terms of functional correctness.

Table 8. Relationships between CK metrics and functional correctness through inverted proxies

Proxy	WMC	DIT	NOC	CBO	RFC	LCOM
FC	-ve	nr	nr	-ve	-ve	-ve
FP	-ve	nr	nr	-ve	-ve	nr
HSF	-ve	nr	nr	-ve	-ve	-
MSF	-ve	nr	+ve	-ve	-ve	-
LSF	-ve	nr	nr	-ve	-ve	-
USF	-ve	nr	+ve	-ve	-ve	-
FD	nc	nc	nc	nc	-	nr
VC	-ve	-ve	-ve	-ve	-ve	-

+ve – positive, -ve – negative, nr – no relationship, nc – no conclusion

Appendix – Selected Studies

- S1. Al-Dallal, J.: Transitive-based object-oriented lack-of-cohesion metric. *Procedia CS*, 1581--1587 (2011)
- S2. Chowdhury, I., Zulkernine, M.: Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities. *Journal of Systems Architecture* 57, 294--313 (2011)
- S3. Al-Dallal, J., Briand, L.C.: An object-oriented high-level design-based class cohesion metric. *Information & Software Technology* 52, 1346--1361 (2010)

- S4. English, M., Exton, C., Rigon, I., Cleary, B.: Fault Detection and Prediction in an Open-Source Software Project. In: 5th International Conference on Predictor Models in Software Engineering (2009)
- S5. Singh, Y., Kaur, A., Malhotra, R.: Empirical validation of object-oriented metrics for predicting fault proneness models. *Software Quality Journal* 18, 3--35 (2010)
- S6. Xu, J., Ho, D., Capretz, L.F.: An Empirical Validation of Object-Oriented Design Metrics for Fault Prediction. *Journal of Computer Science* 4, 571--577 (2008)
- S7. Goel, B., Singh, Y.: Empirical Investigation of Metrics for Fault Prediction on Object-Oriented Software. *Computer and Information Science*, 255--265 (2008)
- S8. Olague, H.M., Etzkorn, L.H., Messimer, S.L., Delugach, H.S.: An empirical validation of object-oriented class complexity metrics and their ability to predict error-prone classes in highly iterative, or agile, software: a case study. *Journal of Software Maintenance* 20, 171--197 (2008)
- S9. Aggarwal, K.K., Singh, Y., Kaur, A., Malhotra, R.: Investigating effect of Design Metrics on Fault Proneness in Object-Oriented Systems. *Journal of Object Technology* 6, 127--141 (2007)
- S10. Pai, G.J., Dugan, J.B.: Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods. *IEEE Trans. Software Eng.* 33, 675--686 (2007)
- S11. Olague, H.M., Etzkorn, L.H., Gholston, S., Quattlebaum, S.: Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes. *IEEE Trans. Software Eng.* 33, 402--419 (2007)
- S12. Gyimóthy, T., Ferenc, R., Siket, I.: Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction. *IEEE Trans. Software Eng.* 31, 897--910 (2005)
- S13. Subramanyam, R., Krishnan, M.S.: Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects. *IEEE Trans. Software Eng.* 29, 297--310 (2003)
- S14. Succi, G., Pedrycz, W., Stefanovic, M., Miller, J.: Practical assessment of the models for identification of defect-prone classes in object-oriented commercial systems using design metrics. *Journal of Systems and Software* 65, 1--12 (2003)
- S15. Emam, K.E., Benlarbi, S., Goel, N., Rai, S.N.: The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics. *IEEE Trans. Software Eng.* 27, 630--650 (2001)
- S16. Emam, K.E., Melo, W.L., Machado, J.C.: The prediction of faulty classes using object-oriented design metrics. *Journal of Systems and Software* 56, 63--75 (2001)
- S17. Briand, L.C., Wüst, J., Daly, J.W., Porter, D.V.: Exploring the relationships between design measures and software quality in object-oriented systems. *Journal of Systems and Software* 51, 245--273 (2000)
- S18. Kamiya, T., Kusumoto, S., Inoue, K., Mohri, Y.: Empirical evaluation of reuse sensitiveness of complexity metrics. *Information & Software Technology* 41, 297--305 (1999)
- S19. Harrison, R., Counsell, S., Nithi, R.V.: An Investigation into the Applicability and Validity of Object-Oriented Design Metrics. *Empirical Software Engineering* 3, 255--273 (1998)
- S20. Basili, V.R., Briand, L.C., Melo, W.L.: A Validation of Object-Oriented Design Metrics as Quality Indicators. *IEEE Trans. Software Eng.* 22, 751-761 (1996)

Acknowledgements. We thank Mohsin Ali, Khalid Wahabi and Mustafa Alsaleh for helping us in literature search, data collection and synthesis. We also would like to thank King Fahd University of Petroleum & Minerals for continuous support throughout this study.

References

1. Chidamber, S.R., Kemerer, C.F.: A Metrics Suite for Object Oriented Design. *IEEE Trans. Software Eng.* 20, 476–493 (1994)
2. Kitchenham, B., Charters, S.: *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. School of Computer Science and Mathematics, Keele University, EBSE Technical Report Version 2.3 (2007)
3. Cohen, J.: *Statistical Power Analysis for the Behavioral Sciences*, 2nd edn. L. Erlbaum Associates (1988)
4. McCabe, T.J.: A complexity measure. *IEEE Trans. Softw. Eng.* SE 2, 308–320 (1976)
5. Hitz, M., Montazeri, B.: Measuring coupling and cohesion in object-oriented systems. *Angewandte Informatik* 50, 1–10 (1995)
6. Henderson-Sellers, B.: *Software Metrics*. Prentice-Hall, Hemel Hempstead (1996)
7. Bansiya, J., Davis, C.G.: A Hierarchical Model for Object-Oriented Design Quality Assessment. *IEEE Trans. Software Eng.* 28, 4–17 (2002)
8. Hopkins, W.G.: Measures of reliability in sports medicine and science. *Sports Medicine* 30, 1–15 (2000)
9. ISO/IEC FDIS 25010, [http://pef.czu.cz/papik/doc/MHJS/pdf/ISOIEC_FDIS25010_\(E\).pdf](http://pef.czu.cz/papik/doc/MHJS/pdf/ISOIEC_FDIS25010_(E).pdf)
10. ISO/IEC 25010 (2011), http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733

Pinpointing Malicious Activities through Network and System-Level Malware Execution Behavior

André Ricardo Abed Grégio^{1,2}, Vitor Monte Afonso²,
Dario Simões Fernandes Filho², Paulo Lício de Geus², Mario Jino²,
and Rafael Duarte Coelho dos Santos³

¹ Renato Archer IT Research Center (CTI/MCT), Campinas, SP, Brazil
`argregio@cti.gov.br`

² University of Campinas (Unicamp), Campinas, SP, Brazil
`{vitor,dario,paulo}@las.ic.unicamp.br, jino@dca.fee.unicamp.br`

³ Brazilian Institute for Space Research (INPE/MCT), S. J. dos Campos, SP, Brazil
`rafael.santos@lac.inpe.br`

Abstract. Malicious programs pose a major threat to Internet-connected systems, increasing the importance of studying their behavior in order to fight against them. In this paper, we propose definitions to the different types of behavior that a program can present during its execution. Based on those definitions, we define *suspicious behavior* as the group of actions that change the state of a target system. We also propose a set of network and system-level dangerous activities that can be used to denote the malignity in suspicious behaviors, which were extracted from a large set of malware samples. In addition, we evaluate the malware samples according to their suspicious behavior. Moreover, we developed filters to translate from lower-level execution traces to the observed dangerous activities and evaluated them in the context of actual malware.

Keywords: computer security, malware analysis, behavioral traces.

1 Introduction

Malicious software are a major threat to Internet-connected systems. This kind of software ranges from worms and trojan horses to rootkits and botnets and is generically referred to as “malware”. Thousands of malware variants arise periodically, hindering their analysis and the creation of effective vaccines by antivirus companies. Publicly available dynamic analysis systems (e.g., Anubis [12], CWSandbox [17], Norman [1], ThreatExpert [2]) provide reports that give an overview of a malware sample behavior. However, they present too many technical details and/or too much information in a slew of activities that may confuse a user on finding the activities that characterize the malignity of an analyzed sample.

We propose a simpler and focused approach to describe malicious activities that is based on the higher-level behavior extracted from analyzed malware samples. Thus, we can bridge lower-level and specialized actions, such as a kernel

function call or a write operation performed into a specific registry key, to understandable, identifiable high-level activities that emphasizes only the suspicious behavior. This can be useful to allow the identification of malware variants, to speed up incident response and to help in the development of malware removal procedures.

The main contributions of this article can be summarized as follows:

- We introduce a new notion of “execution behavior”, splitting it in subsets according to the kind of interference perceived on the target system—active, passive and neutral—modelling a program’s behavior in a simplified way.
- We characterize suspicious behavior by narrowing the scope of malware behavioral analysis to a reduced set of actions that change the state of a system.
- We define a “knowledge base” of network and system-level actions that correspond to intelligible activities (behavioral filters) that extends the set of behaviors described on the field’s literature.
- We developed a prototype tool that is added in our dynamic analysis system to apply the behavioral filters and automatically extract potentially dangerous activities (i.e., suspicious behaviors) from the execution trace of a malware sample.

Additionally, we tested our proposed behavioral filters on a large set of actual malware samples that were gathered from malware collection honeypots [15] and spam attachments and then executed in our dynamic analysis system, BehEMOT [3]. At the end of this process, we pinpoint the malicious activities obtained from these samples and leverage results that allow us to analyze nuances among malware from different sources and with distinct assigned labels.

2 Related Work

In [9], the authors propose a theoretical model to perform behavior-based detection of infectious actions. Their work presents a strong mathematical basis to define different types of behavior and is an extension of a previous work that was only able to handle sequences of bytes [8]. The limiting factor is that their new approach requires the malware’s source code, which is sometimes difficult to obtain.

The authors of [11] describe a malicious behavior model that is based on attribute grammars. They propose an abstraction layer to bridge the semantic gap between the behavior-based detection of malware and subtleties of platforms and systems. They trace a behavior by executing the sample inside a virtual machine and monitoring its system calls, which are then translated to their malicious behavior language. They examine and formally define four types of malware behavior.

It is worth to note that, in our approach, we do not consider the malware sample’s source code, as it would require a decompilation step that could turn into an impossible task, due to the use of packers. Therefore, our work is based

solely on the behavior logged during the execution of a program, which can change due to peculiarities of the monitoring environment [4].

In [14], the authors propose to bridge the semantic gap using behavioral graphs that were manually built to correlate common actions found in malicious bots, such as e-mail sending and data leaking. They evaluate seven kinds of behaviors related to bots, thus their coverage is mostly based on network actions.

Bayer et al. [13] provide a view on different behaviors presented by almost one million malware samples that were analyzed by Anubis over 2007 and 2008. They produced malware statistics and analyze trends, showing the percentage of observed samples that performed a variety of actions, from a simple file creation to the installation of a Windows kernel driver. We took some of their observed behavior to compose our suspicious activity definitions, but instead of analyzing the overall scene we tried to delve into behaviors that we believe that pose more risk to target systems.

Although the literature’s research works are very rich in their definitions and findings, we believe that there is still a lack of focus on the practical applicability of dynamic analysis systems as true generators of heuristics to detection schemes. In addition, previous works on the subject take only specific malware classes into account, e.g., spyware [7], bots [14], worms/viruses [11]. In this paper, we propose to provide high-level behavioral filters to find suspicious behaviors bound to malware samples independently of their assigned classes.

3 Behavioral Traces

To the extent of this work, we use behavioral traces to pinpoint the security-relevant activities that a program performs. In this section, we explain how we extract a malware’s execution trace to identify suspicious activities. Also, we define different types of “behavior” within the scope of this article.

3.1 Extraction and Processing

The first step to extract a behavioral (execution) trace from a malware sample is to run it inside a controlled environment and to monitor the important security-related actions performed during a limited execution time. As the prevalent kind of current malware targets Microsoft Windows-based systems, we chose them as the main focus of interest. Hence, we developed a framework [3] to capture some selected system calls through SSDT hooking [10] and to translate them to high-level actions, logging the produced trace.

The monitored system calls considered in this paper are related to operating system’s entities—file, process and registry—that cover a great variety of potentially suspicious actions and are registered in a trace. This trace is, in its raw form, an ordered set of the performed system calls and their parameters, which need to be processed to represent a higher-level behavior. To monitor a malware sample’s network behavior, we developed a layer driver¹ which is interposed

¹ Layered drivers are kernel modules that can manipulate data flows from the operating system through a specific driver, such as a device driver.

between the network interface driver and the Microsoft Windows operating system. Thus, our layer driver is able to capture all network operations that a malware sample performs during its analysis and that are directed to the drivers that control the devices related to TCP and UDP communication. Therefore, it is possible to obtain network connections and attempts to open local ports made during the analysis time.

To process a behavioral trace we need to translate the monitored system calls into meaningful actions. This is done to facilitate the interpretation of the extracted behavior, as in some cases more than one system call may represent a single operation. Each action is represented by a number of attributes: the **timestamp**, to identify the action's position in the chain of captured events, the **source** process, which represents the performer of an action, the **operation**—i.e., the type of interaction, like `CreateProcess`, `DeleteFile` and `ConnectNetwork`—between the source and the **target** of that operation.

To illustrate this process, let's suppose a program "*mw.exe*" that wants to create a process "*mwproc.exe*". To do this, it calls the `ZwCreateProcess` routine. When this happens, our tool intercepts the system call and produces an action formatted in the following way:

```
<ts>,C:\mw.exe,CreateProcess,C:\mwproc.exe
```

In the same way, the routines `ZwSetValueKey` and `ZwDeleteKey` are translated to the `WriteRegistry` and `RemoveRegistry` operations, respectively.

The advantage of processing system calls in the above way is that when several routines serve to the same purpose, we map them to a single operation. For example, if an action's goal is to delete a file, this can be accomplished by `ZwOpenFile`, `ZwDeleteFile` or `ZwSetInformationFile` with carefully crafted values as their parameters. By abstracting from the particular variant chosen by a monitored program, we are able to present a much more meaningful result, i.e., a `DeleteFile` operation.

3.2 Definitions of Behavior

The general behavior of a program consists of the set of actions performed during its execution by an operating system. In the previous section we defined "action" based on some attributes (timestamp, source, operation, target). Thus, an action " α " is a tuple composed by the values of the aforementioned attributes and can be represented as $\alpha = \{ts, src, op, tgt\}$. Therefore, to define a behavior we proceed as follows:

Let B be the general behavior of a malware sample M_k , and A^{M_k} be the set of N actions α_i performed during its execution, so that $A^{M_k} = \{\alpha_i\}_{i=1..N}$ and $B(M_k) = A^{M_k}$.

The set of actions that compose a behavior can be divided into groups according to their nature: if an action interferes with the environment, i.e. changes the state of the system, it is part of an **active** subset of the behavior. This is the case of actions that involve a file write, delete or creation, for instance. Otherwise,

the action is **passive**, meaning that it gathered a piece of information without modifying anything, for example, read, open or query something.

However, there is a subset of the general behavior that is **neutral**, i.e., the actions can be either active or passive, but they do not lead to a malign outcome. The neutral behavior contains common actions that are performed during a normal execution of any program, such as to load standard system libraries, to read or to configure registry keys and to create temporary files.

3.3 Suspicious Behavior

When a malicious program is executed, each of its actions can be considered suspicious. These actions constitute a suspicious behavior that, when analyzed, may reveal important details related to the attack. For instance, a malware sample that downloads another piece of malicious code and use it to spread itself has to perform a network connection, to write the file containing the malicious code on the compromised system and to launch the process of the downloaded file that will handle the spreading process.

Therefore, we are only interested in actions that modify the state of the compromised system (the active subset of the behavior, that is, B_A) at the same time that we want to avoid the actions that are considered normal to a program's execution (the neutral behavior, that is, B_N). Thus, we define the suspicious behavior of a malware sample M_k as $B_S(M_k) = B_A(M_k) - B_N(M_k)$. From the analysis of each obtained $B_S(M_k)$, we extract a set of network and system-level actions that represent dangerous activities to the security of a system.

4 Malicious Activities

During execution, a software piece interacts actively and passively with the operating system. Thus, benign software presents active behavior such as creating new registries, writing values to registry keys, creating other processes, accessing the network to send debug information or to search for updates, downloading and writing new files etc.

Therefore, as any piece of software does, malware interact with the operating system in the same way. However, malware interactions cause undesired changes on the operating system settings. These changes must be detected to allow for a damage report and to begin an incident response procedure.

Hence, it is necessary to pinpoint the actions that correspond to dangerous or malicious activities, so as to allow better understanding of the malware diversity. To do this, we defined an initial set of network and system-level activities that present a certain level of risk and that can be obtained from selected actions extracted from the suspicious behavior.

4.1 Network-Level Risky Activities

Evasion of Information. Information related to the operating system or the user can be evaded through the network, such as the hostname, hardware data,

network interface data, OS version and credentials. An adversary may use this information to choose targets for an attack, or to map her compromised machines (e.g., zombie computers that are part of a botnet). In a directed attack, sensitive documents may be stolen and transferred to an FTP server, for instance. Also, information can be evaded through a POST (HTTP method) performed on a compromised server, a FTP transfer, an SQL update query to a remote database or an e-mail message sent through an open SMTP server.

Scanning. Worm-like malware need to perform scans over the network to find possible targets for spreading. This involves the search of known vulnerable services or unprotected/open network applications. Apart from spreading, a malware sample may also perform scans to find out a network topology or to find trampoline systems that could be used to launch attacks anonymously.

DoS. There are classes of malware (e.g., bots) whose features include flooding attacks to perform denial of service (DoS). This is oftenly done through the sending of an overwhelming amount of UDP packets, for example, by the nodes (infected machines) of a botnet.

Downloading. Some types of malware are composed by several pieces that execute specialized tasks. Thus, the first piece—the downloader—is responsible for downloading the other components, such as libraries, configuration files, drivers or infected executable files. This compartmentalization is also used by malware developers to try to avoid antiviruses or other security mechanisms. This activity can also indicate a drive-by download, which is a download commonly performed during a user's Web browsing without his/her knowledge.

E-mail Sending. A malicious program can communicate with its owner through e-mail to announce the success of an attack or to send out sensitive data from the compromised machine. Also, a compromised machine can be used as an unsolicited e-mail server, sending thousands of spam on behalf of an attacker that is being paid for the service. The victim's machine is “rented” and acts as a provider of spam or phishing, aiming to distribute commercial messages or even malicious links or attached infected files [6].

IRC Connection. If an attacked system becomes part of a botnet, it needs to “phone home”, i.e., to contact a C&C² server to receive commands, updates etc. Botclients commonly connect to an Instant Relay Chat (IRC) server that acts as a C&C.

4.2 System-Level Risky Activities

Name Resolution File Modification. A trojan-like malware sample can modify the network name resolution file to forward users to a compromised

² Command and Control that manages the bots that belong to a botnet.

server and lure them into supplying their data. These can be credentials (e-mail, online banking, Web applications such as Facebook and Twitter) or financial information (credit card numbers). This kind of modification tricks users to access fake online banking sites as a direct cause of malware known as “bankers”.

Evidence Removal. Some malware disguise themselves as system processes to deceive security mechanisms or forensic analysis: they can “drop” a file that was embedded in a packed way inside their main file or download the actual malicious program from the Internet. In some cases, these droppers/downloaders remove the evidence of compromise, deleting the installation files after the attack. In addition, a malware sample that is able to identify that it is being analyzed may also remove itself from the system.

Critical Registry Key Removal. There are registry keys that are critical to the normal operation of a system, e.g., the one that allows initialization in secure mode. The removal of this kind of key can cause instability in the system and inconvenient obstacles during a disinfection procedure.

Security Mechanisms Corruption. To compromise a target system while avoiding detection, malware authors usually try to identify and disable security mechanisms. This activity can be accomplished by turning off the system firewall or known antivirus engines, through the termination of their processes and/or removal of the associated registry keys.

Browser’s Proxy Modification. The effect of this activity is similar to that described on “name resolution file modification”. The difference here is that a malware sample loads a configuration file in the browser’s memory (when it is running) that changes the proxy on the fly, resulting in an automatic redirection of the user to a malicious site. Malware usually do this using PAC (proxy auto-config) files, which are effective on different operating systems and browsers.

Driver Loading. Drivers are kernel modules that access the most privileged level of a system. A driver makes the interface between the operating system and the hardware, such as network interfaces, graphic cards and other devices. However, drivers are also used by rootkits, a kernel-level kind of malware that can hide their processes, files and network connections in order to remain undetected.

5 Experimental Results

We collected 1,641 malware samples from July, 2010 to July, 2011—463 from honeypots (**collector**), 1,182 from spam messages (**phishing**)—and executed them in our dynamic analysis environment, which is a Qemu-emulated [5] MS Windows XP SP3. This execution produced a behavior trace for each analyzed sample, which were also sent to the VirusTotal service (<http://www.virustotal.com>) so that we could get their Kaspersky Anti-Virus (KAV) label. We then applied the

behavioral filters described in Section 4 to the traces and analyzed the network- and system-level malicious activities. We discuss, the observed malicious activities over the full malware set in Section 5.1 and the results regarding different malware classes (attributed by KAV) in Section 5.2

5.1 Malicious Activities' Pinpoint

The purpose of the behavioral filters is to map suspicious actions performed by a program to intelligible activities. These filters provide high-level and useful information about a malware sample execution and describe its presented behavior. For example, if a malware sample tries to turn off the security mechanisms natively running on a Windows OS to avoid detection and weaken the machine defenses, it commonly launches a script that performs some shell commands, such as `net stop "Security Center"`, `net stop SharedAccess` and `netsh firewall set opmode mode=disable`. Also, a sample might perform changes on the `FirewallPolicy\StandardProfile` registry keys, for instance, by setting the value of `EnableFirewall` to "0".

This kind of action causes a positive match against our behavioral filter and leverages, in the particular aforementioned example, "Security Mechanism Corruption" as a malicious activity found in the evaluated sample. When the "pinpointing" process is finished, we have a list of dangerous (and potentially malicious) activities for each sample from our malware dataset. This process produced the results from Table 1, divided by source (phishing or collector), when applied to the complete dataset.

Table 1. Malicious activities discovered through the pinpointing process on our collection; sum may be higher than 100%

Level	Activity	Phishing (%)	Collector (%)
Network	NT1 (Evasion)	3.72	6.69
	NT2 (Scan)	14.21	50.54
	NT3 (DoS)	37.22	29.37
	NT4 (Download)	1.10	9.07
	NT5 (E-mail)	1.95	3.45
	NT6 (IRC)	0.42	9.28
System	OS1 (Hosts File)	1.10	0.43
	OS2 (Evidence)	15.06	4.32
	OS3 (Critical Key)	0.34	0.43
	OS4 (Security Bypass)	4.48	5.18
	OS5 (PAC)	0.25	0
	OS6 (Driver)	5.16	0

We notice that most of the analyzed malware samples performed the same set of malicious activities, despite their source. Those activities are attempts to scan networks for vulnerable services or UDP flooding (NT2, NT3) at the

network-level and self-removal and security mechanism bypass (OS2, OS4) at the system-level. From the samples that came from our collectors, 15.15% did not present any behavior, either due to crashing during the execution, to corrupted binaries or to the use of anti-analysis techniques. From the samples obtained by e-mail crawling (phishing set), 12.01% either presented an incomplete trace or did not match any of our defined suspicious behaviors.

5.2 Malware Classes Behavior

As mentioned previously, we obtained the KAV labels from VirusTotal for each malware sample from our dataset. Then, we processed these labels to extract only the assigned class (e.g., trojan, worm, backdoor etc) according to the Kaspersky naming rules [16]. These rules define a naming system that is composed by `[Prefix:]Behavior.Platform.Name[.Variant]`, where the parameter *Behavior* represents the malware class.

Hence, we grouped those samples whose assigned class is the same and analyzed the ten more populated classes, which correspond to more than 85% of the samples (excluding the $\approx 7\%$ that are unknown to antivirus engines at the time of this analysis, i.e., August, 2011). After that, we tested our behavioral filters on each sample of the ten selected classes to extract their malicious activities (defined in Section 4).

We show the network-level malicious activities performed by the different classes in Table 2 and the system-level ones in Table 3, where a checkmark (✓) denotes that at least one of the samples assigned to the class (rows) performed the suspicious activity (column) and a blank entry denotes that this behavior was unmatched.

Table 2. Union of network-level risky activities per malware class

Class	NT1	NT2	NT3	NT4	NT5	NT6
Worm	✓	✓	✓	✓	✓	
Backdoor	✓	✓	✓	✓	✓	✓
Trojan	✓	✓	✓	✓		
Downloader	✓		✓	✓	✓	
Virus		✓	✓			
UNKNOWN		✓	✓			
Packed			✓			✓
Gamethief			✓			
Banker	✓		✓		✓	
Dropper	✓		✓		✓	

These tables are the union of the pinpointed malicious activities from a specific AV-assigned class, i.e., if at least one sample from the assigned class performed the activity, then we put a checkmark. The ideal situation happens when a malware class is characterized by a specific behavior, for instance, a downloader

Table 3. Union of system-level risky activities per malware class

Class	OS1	OS2	OS3	OS4	OS5	OS6
Worm	✓	✓	✓	✓		✓
Backdoor	✓	✓		✓		✓
Trojan	✓	✓	✓	✓		✓
Downloader	✓	✓		✓		✓
Virus		✓		✓		
UNKNOWN		✓		✓		✓
Packed		✓		✓		✓
Gamethief		✓		✓		✓
Banker	✓			✓	✓	
Dropper	✓	✓		✓		✓

obtains something from the Internet and a worm tries to spread. Although this may be true, antivirus labels are not good to separate malware in meaningful, representative classes as the assigned name can confuse and mislead the user about the actual behavior of a sample.

Thus, if we take a closer look on the tables' results, it is worth noting that there are classes whose samples share a great amount of suspicious activities among each other. To illustrate this, let's analyze the three most populated classes: worm, backdoor and trojan. From Table 2, it seems that worms differ from backdoors only by "NT6", whereas trojans differ from worms by "NT5" and from backdoors by "NT5" and "NT6". However, due to the fact that our results are presented as the union of identified suspicious activities, there could be samples from these three distinct classes that share a common subset of network and system-level presented behavior. One such instance might happen when some samples from the worm, backdoor and trojan classes perform exclusively the suspicious network-level activities "NT1", "NT2", "NT3" and system-level activities "OS1" and "OS2". Therefore, although these samples are classified by KAV into three distinct classes, if we consider their observed behavior they should be assigned to a single one. Unfortunately, all antivirus engines share the same problem, making their malware assigned labels nearly useless to users when regarding the malicious behavioral information.

Conversely, our approach produces detailed enough information that can provide a better understanding about the risks related to a program's execution. It is also possible to overcome the misclassification of antivirus engines by classifying the unidentified (UNKNOWN) samples by their traced behavior. This way, our scheme can be used to generate a malware class characterized by "NT2", "NT3", "OS2", "OS4" and "OS6", thus avoiding the false-negative results produced by antivirus engines.

It is interesting to notice that in Table 3, as expected from malware that attack online banking sites, only the bankers presented the behavior labeled as "OS5" (Browser's Proxy Modification).

6 Conclusion

In this paper, we divided a program's execution trace in different types of behaviors and proposed the suspicious behavior definition to denote the dangerous activities that change the state of a system. We leveraged behavioral filters composed by these activities—performed at the network and system-level—to identify potentially harmful actions and to help with incident response as well as to provide a better understanding of malware. To evaluate our approach, we tested it in a dataset of malware collected from different sources. We provided results that show the percentage of malware samples that presented our behaviors and that compare them to AV-assigned classes. This latter comparison pointed to the problems in the current malware naming scheme, which we plan to address in a future work.

References

1. Norman Sandbox, http://www.norman.com/security_center/security_tools/
2. ThreatExpert, <http://www.threatexpert.com/>
3. Afonso, V.M., Filho, D.S.F., Grégio, A.R.A., de Geus, P.L., Jino, M.: A hybrid framework to analyze web and os malware. In: Proceedings of the 2012 IEEE International Conference on Communications (ICC) (June 2012)
4. Balzarotti, D., Cova, M., Karlberger, C., Kruegel, C., Kirda, E., Vigna, G.: Efficient detection of split personalities in malware. In: 17th Annual Network and Distributed System Security Symposium, NDSS 2010 (February 2010)
5. Bellard, F.: Qemu, a fast and portable dynamic translator. In: USENIX Annual Technical Conference, FREENIX Track, pp. 41–46 (2005)
6. Calais, P.H., Pires, D.E.V., Guedes, D.O., Meira, W., Hoepers, C., Steding-jessen, K.: A campaign-based characterization of spamming strategies. In: Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS) (2008)
7. Egele, M., Kruegel, C., Kirda, E., Yin, H., Song, D.: Dynamic Spyware Analysis. In: Proceedings of the USENIX Annual Technical Conference. USENIX Association, Berkeley (2007)
8. Filiol, E.: Malware pattern scanning schemes secure against black-box analysis. *Journal in Computer Virology* 2(1), 35–50 (2006)
9. Filiol, E., Jacob, G., Le Liard, M.: Evaluation methodology and theoretical model for antiviral behavioural detection strategies. *Journal in Computer Virology* 3(1), 23–37 (2007)
10. Hoglund, G., Butler, J.: *Rootkits—Subverting the Windows Kernel*. Addison-Wesley (2006)
11. Jacob, G., Debar, H., Filiol, E.: Malware Behavioral Detection by Attribute-Automata Using Abstraction from Platform and Language. In: Kirda, E., Jha, S., Balzarotti, D. (eds.) RAID 2009. LNCS, vol. 5758, pp. 81–100. Springer, Heidelberg (2009)
12. Kruegel, C., Kirda, E., Bayer, U.: TTAalyze: A tool for analyzing malware. In: Proceedings of the 15th European Institute for Computer Antivirus Research (EICAR 2006) Annual Conference (April 2006)
13. Kruegel, C., Kirda, E., Bayer, U., Balzarotti, D., Habibi, I.: Insights into current malware behavior. In: 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET), Boston (April 2009)

14. Martignoni, L., Stinson, E., Fredrikson, M., Jha, S., Mitchell, J.C.: A Layered Architecture for Detecting Malicious Behaviors. In: Lippmann, R., Kirda, E., Trachtenberg, A. (eds.) RAID 2008. LNCS, vol. 5230, pp. 78–97. Springer, Heidelberg (2008)
15. Provos, N., Holz, T.: Virtual honeypots: from botnet tracking to intrusion detection, 1st edn. Addison-Wesley Professional (2007)
16. SecureList: Rules for naming detected objects,
<http://www.securelist.com/en/%20threats/detect?chapter=136>
17. Willems, C., Holz, T., Freiling, F.: Toward Automated Dynamic Malware Analysis Using CWSandbox. IEEE Security and Privacy 5, 32–39 (2007)

A Malware Detection System Inspired on the Human Immune System

Isabela Liane de Oliveira¹, André Ricardo Abed Grégio²,
and Adriano Mauro Cansian¹

¹ São Paulo State University (Unesp), São José do Rio Preto, SP, Brazil
{isabela,adriano}@acmesecurity.org

² Renato Archer IT Research Center (CTI/MCT), Campinas, SP, Brazil
andre.gregio@cti.gov.br

Abstract. Malicious programs (malware) can cause severe damage on computer systems and data. The mechanism that the human immune system uses to detect and protect from organisms that threaten the human body is efficient and can be adapted to detect malware attacks. In this paper we propose a system to perform malware distributed collection, analysis and detection, this last inspired by the human immune system. After collecting malware samples from Internet, they are dynamically analyzed so as to provide execution traces at the operating system level and network flows that are used to create a behavioral model and to generate a detection signature. Those signatures serve as input to a malware detector, acting as the antibodies in the antigen detection process. This allows us to understand the malware attack and aids in the infection removal procedures.

Keywords: malicious code, human immune system, data mining.

1 Introduction

Malicious programs (malware) are one of the main threats to the security of Internet users. Venues that can be used in malware attacks include specially crafted email containing malicious attachments or links to compromised sites and even the simple Internet browsing process. When a computer system is attacked by malware, it has no integrity anymore as sensitive data can be modified, stolen, shared or exposed, including credentials, passwords, documents and financial information. Also, an infected system can be used to attack other networks and hosts, or to serve as storage to illegal content.

A malware piece can be defined as a program whose instructions perform actions that are harmful to a system, once they are executed. There are many classes of malware [2], such as viruses, worms, trojan horses, backdoors, screenloggers, rootkits and bots, which may act individually or in association in order to combine features. Their goal is to compromise a system, to take the control of it, to attack other devices, to capture user information and to use the system resources while hiding from security mechanisms or trying to disable or subvert them.

Thus, the observed malware behavior can be compared to the antigens (substances capable of inducing an immune response) behavior present in the human immune system. As well as it occurs with diseases, new samples of malware may arise or existing ones can evolve and become stronger, turning the existing protections ineffective. Therefore, security mechanisms must be constantly updated to provide acceptable and timely protection.

In this paper, we propose a system to collect, to analyze and to detect malware that is inspired on the features of the human immune system. The goals of this system are to "learn" how new malware samples act through executing them in target systems and to store this knowledge as signatures in a database to produce countermeasures. The main contribution of this paper is, besides the proposed design based on the human immune system, the automated signature modelling that is used to detect malware, which combines the behavior shown in the victim's operating system and the network traffic captured.

The remaining of the paper is divided as follows: In the Section 2 we show some basic concepts that are need to an adequate comprehension of our work. Also, there is a summary of some important related works. In the Section 3, we describe our proposed system, the signature modelling steps and the detection process. In the Section 4, we discuss the tests performed and the obtained results. Finally, in the Section 5 we present the concluding remarks.

2 Concepts and Related Work

In this section, we briefly present some features from the human immune system that inspired us to design the proposed system, as well as the basic concepts regarding network flows, which are used as part of the detection signatures, and data mining, which is applied to create the profiles used on the detection scheme.

2.1 Human Immune System

The human immune system is able to ensure the survival of an individual throughout his life, even when exposed to potentially deadly bacteria and viruses everyday. Thus, the biological system provides a rich source of inspiration for the maintenance of computational security [15].

The human immune system [6] has three layers: the anatomic barrier, the innate immunity and the adaptive immunity. Those layers together provide the defense of the human body. The anatomic barrier consists of the skin and the surface of mucous membranes and forms the first layer of biological defense. The innate immunity has an inborn nature and a limited ability to differentiate pathogens, reacting similarly against most infectious agents. Finally, the adaptive immunity is able to specifically recognize a pathogen agent, thus allowing fairly efficient responses. Another interesting feature of the human immune system is its ability to store the information gained about an infectious agent, in order to respond more vigorously to new exposures to this agent.

The main function of the immune system is to recognize and to respond to substances known as antigens, which are able to produces an immune response. To do this response, the system must perform pattern recognition tasks to distinguish the body’s cells, called **self**, from foreign substances, called **nonself**. This classification corresponds to the negative selection process.

2.2 Network Flow

A data flow in the Netflow standard (also known as a network flow) is defined as a unidirectional sequence of packets exchanged between unique source and destination hosts [4]. A network flow provides an information summary about the traffic that passed on a specific network device (router or switch). Thus, flows are tuples in which the following information have the same value: source and destination IP address; source and destination port; protocol field value from the IP datagram; Type of Service byte value from the IP datagram and the logical input interface in the router or switch in which the collection occurs. These fields allow that a flow represents concisely the traffic through a given point of observation (e.g., router). All datagrams with the same value in the fields of their tuples are counted, grouped and packed in a flow record. A scheme representing the flow collection and the network information captured can be seen in Figure 1.

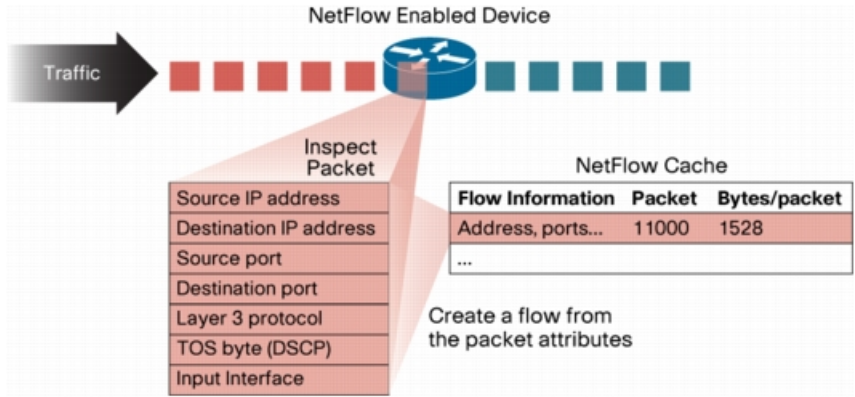


Fig. 1. Scheme of Cisco Netflow standard of data flow [3]

The flows that are generated in the collection device are stored and exported, making it a valuable source of information. It is possible to obtain details from each connection and by any machine that belongs to the monitored environment, this being highly relevant to traffic analysis.

In [5], the authors discuss a new architecture for event detection in computer networks using the NetFlow protocol, storing the information flows in a database. The storage part of the system proposed in this paper is based on this referred architecture.

2.3 Data Mining

Data mining, in general, consists in discovering knowledge in large amounts of data (Knowledge Discovery in Databases or KDD) [11]. However, these knowledge may not be previously unknown and the data to be analyzed need not to be organized in databases [10]. The process of knowledge discovery is summarized basically in three steps:

- **Pre-processing:** the data is selected (according to its importance) and prepared (removal of noise and irrelevant data) to serve as input to the next step.
- **Data mining:** applies some statistical techniques or artificial intelligence in order to extract useful information.
- **Analysis of results:** the results are interpreted, for example, to determine some additional knowledge or to define the importance of the produced events.

A data mining process can perform at least one of the following methods: Class Description, Association, Classification, Prediction, Clustering and Time Series Analysis [11]. To the extent of this paper, we use the Classification method to analyze a training data set and then to build a model for each class based on the data features. Thus, we chose a decision tree, i.e., a rule set produced in a classification step and that can be used to understand better the data and to classify future instances. The motivation behind this is that decision trees are easy to understand and they can be automatically recreated as new data or phenomena appear.

2.4 Related Works

One of the most important research works related to the application of immunology concepts to the security area is the one from Forrest et al. [9]. In their work, the authors describe a method to distinguish a legitimate user from an unauthorized user on a system based on the "T" cells generation performed by the human immune system. Kephart [14] proposes a system that develops antibodies against computer viruses, identifying them in a way that is also inspired on the biological immune system. Kephart noted that the immune system has some important properties that can be deployed as a decentralized system whose purpose is the identification of computer viruses.

In [12], the authors discuss about scalability and coverage problems of the AIS (Artificial Immune System). The AIS uses negative selection algorithms (NSA) to perform malicious code detection through the difference between normal states (self) and unknown substances (nonself). However, it requires a large number of detectors to avoid a large amount of false negatives and the training of these detectors demands a long time. To solve this problem, the authors propose a model entitled "Collaborative Artificial Immune System". In their model,

independent immune bodies in different computers are organized in a virtual structure named “Immune Collaborative Body” (ICB). The ICB allows the detectors’ sharing to improve the detection’s efficiency. A collaboration module is added to each immune body to communication and coordination. This model is based on the fact that certain malware classes, such as worms, try to spread in similar computer systems due to the increased probability that they have the same vulnerabilities.

In [1], the authors propose a flow analysis and monitoring system based on the Netflow standard to attend enterprise networks. This system is divided into three modules:

- Collection and storage of NetFlow flows.
- Web interface for viewing the stored data.
- IDS (Intrusion Detection System) that provides monitoring for anomalous traffic through two statistical algorithms—one based on variance similarity and another one based on the Euclidean distance—and the detection of some malware types and network attacks through a pattern matching algorithm based on the descriptions of events using data flows.

Their system only detects two malware types: trojan horses and worms. The detection is performed based on the features obtained from the flows and they lack the definition of a signature model. Their paper does not discuss how these features are extracted or how the features of new malware are inserted into their database.

A new trend related to malicious botnets is the use of alternative communication channels, such as DNS-tunnelling and HTTP instead of IRC, between the command and control (C&C) servers and infected hosts. Based on this fact, in [16], the authors propose a system to detect botnets that use fast-flux domains using Domain Name Service (DNS) queries. They use C5.0 decision tree classifier and Naive Bayesian classifier to classifying a domain as fast-flux or legitimate. Furthermore, the system analyze textual features of domain names to detect algorithmically generated domain names through Naive Bayesian, Bayesian, Total Variation distance and Probability distribution. The best results were obtained with Naive Bayesian and had low false positive rates. The results with Total Variation Distance classifier and Probability classifier were similar to Naive Bayesian. However, the C5.0 decision tree results shown some limitations and they recommended her use in conjunction with other detection techniques.

Another work involving malicious code detection through NetFlow flows is [13], in which the authors propose a malicious website detection system. In this case, a dataset of normal and malicious web traffic that is collected by a NetFlow collector is used to create spatial-temporal aggregating variables. These flows are used by three classification learning methods (Nave Bayesian, Decision Tree and Support Vector Machine) to build a prediction model that distinguishes between normal and malicious web traffic. The prediction model generated in the training step is then used to perform malware detection in real time. Their best results were obtained using decision trees, but this is not deeply discussed.

3 Proposed System

3.1 Architecture

Computer system interact with several threats available on the Internet as the human body is exposed to many antigens. Inspired by this fact, we propose a modular system to collect, to store, to analyze and to detect malware using ideas from the human immune system.

Our proposed architecture, as a whole, can be divided into three systems, distributed collection system, malware dynamic analysis system and network monitoring system, which can be seen in Figure 2.

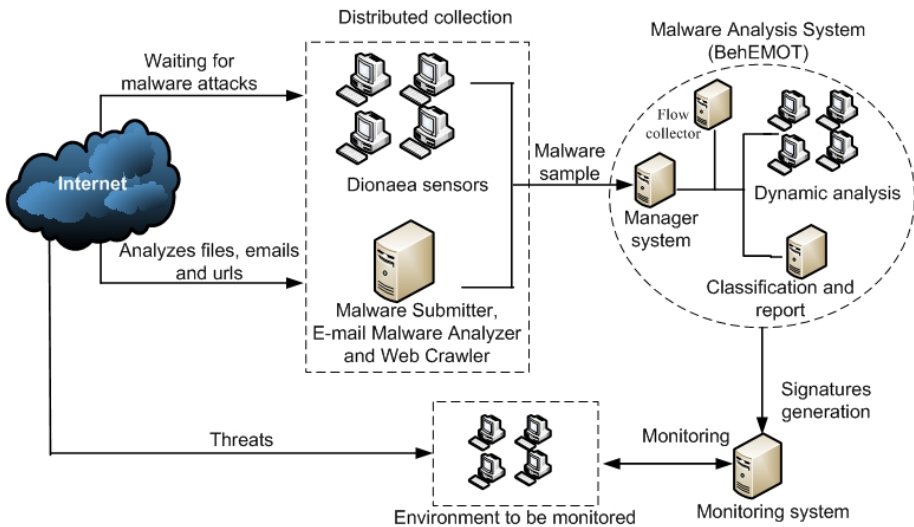


Fig. 2. Modular architecture for our proposed system

The distributed collection system has two subsystem. The first one contains several sensors that emulate vulnerable network services so that Internet malware can exploit them. To do this, we installed Dionaea honeypots that collect all malware samples that can successfully complete an attack [7]. The second subsystem is a web interface composed by the following tools: E-mail Malware Analyzer (EMA), Web Crawler (WC) and Malware Submitter (MS). EMA looks for malicious codes present on links or attached to email messages, whereas WC is responsible for tracking specific sites on the Internet that may be malware-infected. Finally, MS analyzes whether the submitted file is malware or not. All collected samples are sent to a dynamic analysis system, which is explained below.

The dynamic analysis system is divided into two modules: management and analysis. The management module is responsible for receiving malware samples and for controlling the analysis queue and the data produced during the monitoring of a sample's execution so as to generate reports about the malware behavior observed. In the analysis module, we dynamically analyze a malware sample in an automated fashion. This analysis produces information about the sample's execution behavior, i.e., the interaction between a malware piece and the target system. To do the dynamic analysis we use BehEMOT [8], a tool that monitors the specific malware actions performed at the operating system's kernel level, such as creation, deletion or changes related files and registry keys, processes created or terminated, mutex operations and network traffic. From this monitoring, BehEMOT creates an analysis report summarizing the captured behavior, which is based on the system calls that the malicious process performed. This report is then used to generate an execution profile of the analyzed malware sample.

To the BehEMOT basic environment we added a data flow collection module that is used to characterize the network traffic produced by malware. These flows pass through a negative selection process, which generates a network signature. The system calls obtained by dynamic analysis also pass through a negative selection process, from which a behavior signature is extracted.

These two types of signatures are stored in a database and its purpose is to act as the antibodies to the monitored environment. Thus, the monitoring system proposed uses the signatures to perform the detection of a malware sample that is threatening it. If a malware threat is detected, the immune response is triggered and an alert message is sent to the monitored environment administrator.

3.2 Signature Generation System

The signature generation system is illustrated in Figure 3. This figure represents an analogy between the malware signature generation process and the human immune system. To generate a consistent signature, a malware sample is executed three times in the BehEMOT system. As the malware sample acts as an antigen for the human immune system, its execution represents the process of acquiring information about the object that causes damage, in this case, to a computer system. Thus, for each of the three executions performed, the network outgoing traffic from the target machine (as data flows) and the system calls used by the malware process are stored to be analyzed in the next step.

After this information collection step, the negative selection process is applied to the stored data and the irrelevant data is discarded. To remove the noise, we extract only those features that appear on all the three executions. Those common information is then transformed into a signature. In the end of this maturation process, a network signature and another behavior signature that characterize the malicious program are generated and stored in a database. This learning process aggregates "memory" (history) to the system, as future occurrences of this program in a monitored environment may be detected.

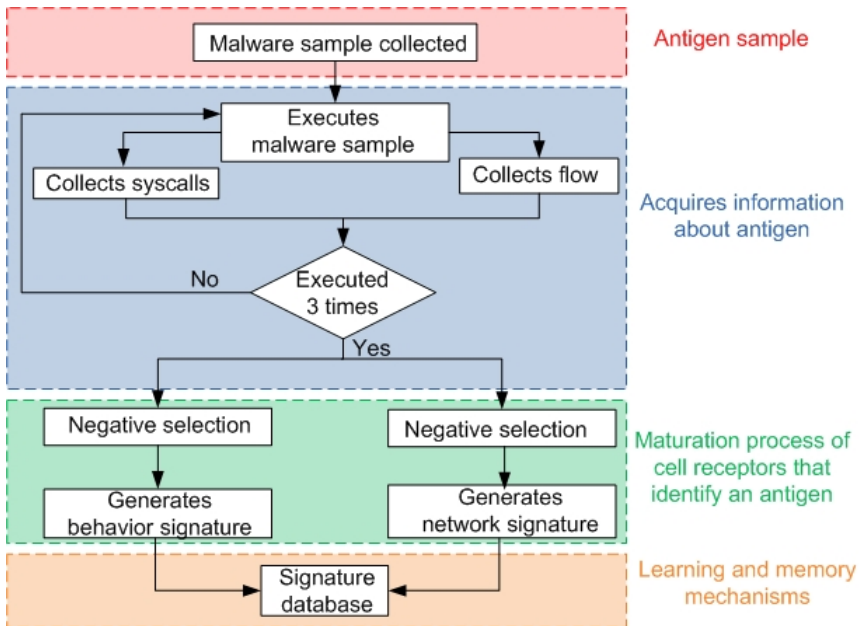


Fig. 3. Signature generation process

3.3 Signature Model

As said in the previous section two types of signatures are developed: behavior and network. The behavior signature model is created based on information produced by malware dynamic analysis. A malware behavioral profile is pre-processed in a way that, for each activity captured, a tuple is generated with the

Table 1. Excerpt of behavior signature from a malware sample

Action	Action Type	Action Source	Action Target	Parameters
WRITE	REGISTRY	%HOMEPATH%\desktop\MALW RE.exe	\registry\machine\software\micro soft\windows\current version\internet settings\cache\paths\path	
[...]	[...]	[...]	[...]	[...]
CREATE	FILE	%HOMEPATH%\desktop\MALW RE.exe	%PATH%\nt09.exe	
[...]	[...]	[...]	[...]	[...]
WRITE	REGISTRY	%PATH%\nt09.exe	\registry\user\{RID}\software\micro soft\windows\currentversion\explorer\shell folders\cookies	

following fields: action (eg, create, delete), action type (eg, file, process, register, network, mutex), action source (executor), action destination (target) and action parameter, if any. An example of behavior signature is shown in the Table [11](#)

We create the network signature model based on the data that is collected from a user's machine network flows. A flow generated by a malware sample passes through a pre-processing step that generates a tuple. This tuple consists of the following information, which is extracted from the infected machine outgoing traffic: total bytes, destination port number and transport layer protocol, as illustrated in Table [2](#). The network signature is also composed of one or more tuples that are extracted from the collected flows.

3.4 Monitoring System

To perform the detection of a malicious program running on the monitored environment, we developed an agent that analyzes system calls performed on the machine this agent is installed and then compare them to the behavioral signatures stored in our database. If there is a match, a warning message is sent to the environment administrator to inform the detection of a malware threat. These captured system calls also pass through a negative selection process, which is made in a pre-processing step. This way, only those system calls that are relevant from the system's security point of view are compared to a signature present on our database.

Another agent was developed to analyze the network flows captured on the monitored environment and to compare them to the network signatures stored in the database. Here, the process is the same mentioned earlier, incurring in the alert message being sent. Figure [4](#) illustrates the monitoring of an environment.

Due to the high amount of flows that a network can generate, we chose to use a decision tree within the network agent to classify flows as **normal** (not caused by a malware) or **suspicious** (may have been caused by a malware). Thus, only those considered as suspicious are compared with the signatures. In Figure [5](#), we illustrate the process to create a decision tree that will be used by the network agent. We used both normal and suspicious data flows (network signature) as input for the training and the creation of the decision tree classifier. The normal flows passed through processing in which flows were transformed into tuples in the same format of those from the network signatures. Then, we used the WEKA framework [18](#) to perform the data mining process using the input data and creating a decision tree. In a later step, this decision tree is included within the network agent to serve as the flow classifier. A decision tree generated for this purpose is illustrated in Figure [6](#).

To train the decision tree, we used 268,853 tuples, being approximately 1.02% suspicious tuples and 98.98% normal data. This can be justified by the fact that we expect that the majority of traffic is legitimate (normal dns queries, e-mail messages, navigation and transfers) and that a small fraction is related to malicious content (scans, malware spreading etc). The time spent to build the decision tree was approximately 3.31 seconds and the time to test the model was approximately 1.09 seconds. The training results showed that 99.99% of the input

Table 2. Excerpt of a network signature from a malware sample

Total Bytes	Destination Port	Protocol
61	53	UDP
2136	80	TCP
[...]	[...]	[...]
106	4244	TCP

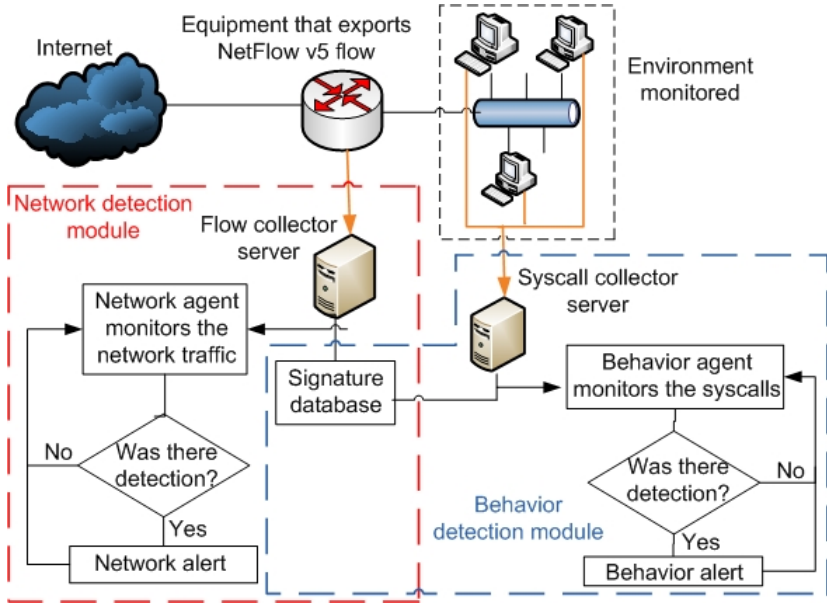


Fig. 4. Monitoring System

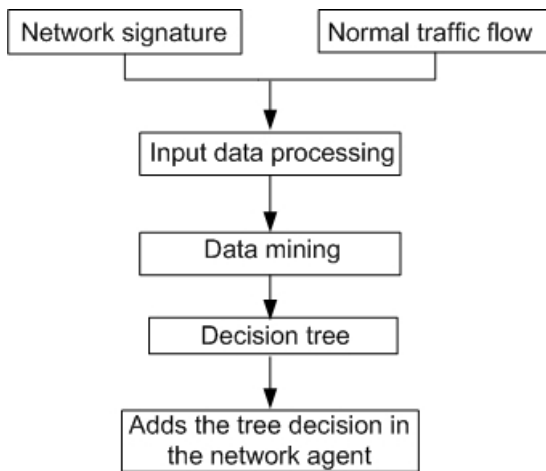


Fig. 5. Process to create the decision tree used by the network agent

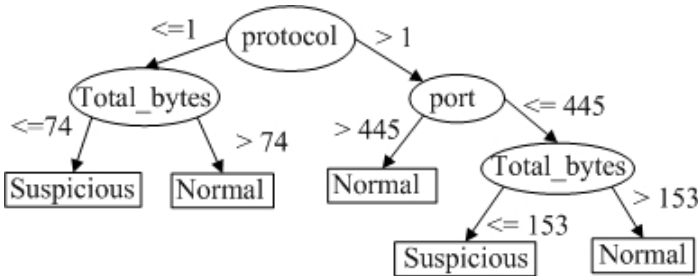


Fig. 6. Example of decision tree used as a classifier for network flows

data were correctly classified and 0.01% were false positives, i.e., normal data that were misclassified as suspicious. These training results show the benefits of using decision tree, due to its efficiency, effectivity and ease to deploy (a chain of if-then-elses). This way, the detection can be performed in a fast way and with a low complexity.

Both behavior and network detectors can send to the administrator of the monitored environment an alert that is composed by a general information report about an identified malware, as previously mentioned. Moreover, the network detector also sends the IP address of the suspect machine. Figure 7 illustrates an example of a report from an alert message.

4 Tests and Results

The proposed distributed collection system was deployed in two separate networks inside geographically apart institutions. All malware collected using our sensors are stored to be furtherly sent to our dynamic analysis system. The average quantity of unique malware samples that were collected per month (from January 1 to December 31, 2011) in one of the sensors was 53. The second sensor had an average of 23 unique malware samples collected per month between March 1 and December 31, 2011.

Figure 8 illustrates the report displayed by the web interface (a subsystem of the distributed collection system) when a malware sample is detected by one of the following tools: EMA or Web Crawler or Malware Submitter. The output of the report's session "Detection" shows the labels assigned by several antivirus engines and is summarized in the aforementioned Figure.

In the next subsection we show the obtained detection results.

4.1 Detection Results

After the signature model definition, the collected malware samples were submitted to the subsystem responsible for the signature generation process. All signatures that are generated in this step are stored in a database and serve as input for the detection provided by the agents installed in the monitored system.


```
##### DETECTION ALERT #####
ID: 44
IP: 192.168.0.2
Name: 0a367bfd7f4ddfd24db6409126ece748
Creation date: 2011-12-24
File Type: PE32 executable for MS Windows (GUI) Intel 80386 32-bit
Avira: N/I
MD5: 0a367bfd7f4ddfd24db6409126ece748
SHA1: a4ab9f3ded2e5d8590b5775668ee5b9328d18c9d
SHA256:
7e25127f9582285d72866ca7c926f1df9728e7762fb8d4529e4c569d77a1ab9e
Packer: Nenhum packer encontrado
To remove the malware:
1. Boot your computer into safe mode.
2. Back up your system before making any changes.
3. Remove the following files:
- %PATH%\nt09.exe
[ ... ]
4. Open Registry Editor and delete the following registry entries:
- \registry\user\{RID}\software\microsoft\windows\currentversion\explorer\shell
folders\cookies
[ ... ]
- \registry\machine\software\microsoft\windows\currentversion\internet settings
\cache\paths\path1\cachepath
5. It is possibly for malware to load by hiding within the system WIN.INI file and the
strings "run=" and "load=". You must check it and remove it from your computer.
Note: Some path file or registry can change
```

Fig. 7. Example of an alert report from a network detector

```
Suspiciou file: Backdoor.IRC.Kelebek.h
MD5:          f4f05ef10b6f6d285c124d3ad738b29b
Type:         PE32 executable for MS Windows (GUI) Intel 80386 32-bit
              Emsisoft:          Backdoor.IRC.Kelebek!IK
              Comodo:            Backdoor.Win32.IRC.~B
              F-Secure:         Backdoor.Irc.Kelebek.H
Detection:    DrWeb:            Trojan.MulDrop3.6237
              Rising:           Trojan.Win32.Generic.122B54D0
              Ikarus:           Backdoor.IRC.Kelebek
              Fortinet:         IRC/FLOOD.DT!tr.bdr
              AVG:              IRC/BackDoor.Flood
              Panda:            Trj/Dropper.WF
Packer:       Install Stub 32-bit
```

Fig. 8. Example of report produced by the distributed collector system when a malware is detected and displayed through the developed Web interface

The 1,021 collected malware samples that produced a signature were submitted to Virus Total [17], an online service whose aim is to analyze suspicious files and URLs by scanning them with a variety of antivirus engines in an attempt to identify already known malware. Figure 9 illustrates the total amount of malware samples that were detected by some of the available antiviruses. In the cases where the Virus Total service did not show any result from a specific antivirus engine, we considered the malware sample as undetected.

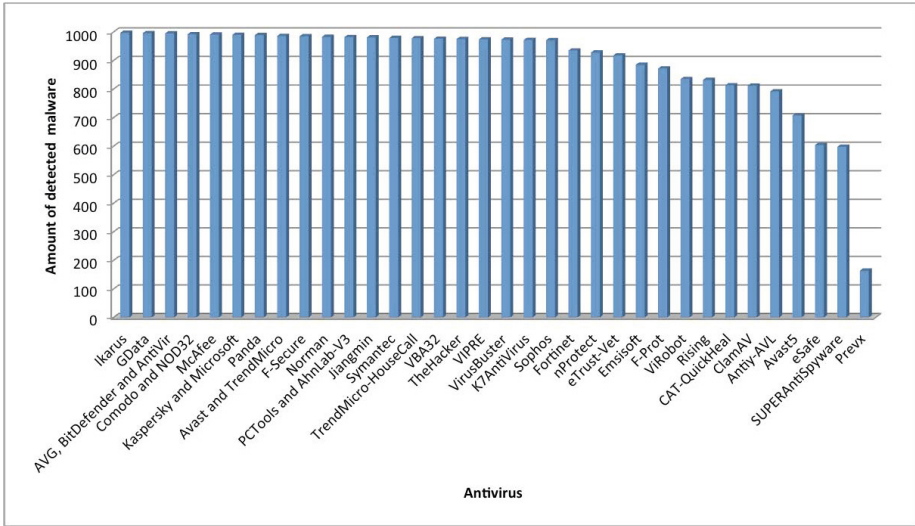


Fig. 9. Chart showing the detection rate of each antivirus engine

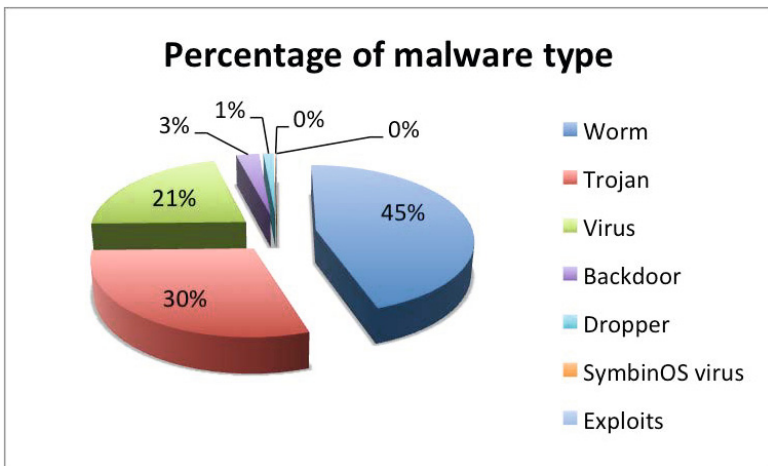


Fig. 10. Percentage of malware (by type) with signature

As we can observe in the figure above, the antivirus with the higher detection rate is Ikarus (97.65%) and the antivirus with the worst rate is Prevx (16,06%). Figure 10 illustrates the malware amount according to their type. Note that the most frequent malware classes are Worm, Trojan and Virus.

5 Conclusion

The threat that a malicious software poses to computer systems, which increases by the vast amount of variants that arise on a daily basis, demands the development of new defense mechanisms and protection methods. In this paper, we proposed a distributed system to collect, analyze and detect malware based on their execution behavior on a target operating system and their produced network traffic (in the form of network flows), which is inspired by the human immune system.

The main advantage of the immune system is its adaptability in face of new problems and its knowledge gain (memory), which provides a history that helps to handle already known threats. To populate a database responsible for this memory, we analyzed hundreds of malware samples that we have collected in our system, from which we have extracted network and behavior signatures. This gained knowledge is then used to detect attacks in actual monitored environments. The detection is performed by agents that we developed, whose function is to monitor the network and operating system activities in an environment that must be protected. If a malware is detected the agent sends an alert email message.

In Table 3, we summarize the analogy between the human immune system and our proposed system, explained in more details below:

- The malware pieces that threaten computer systems are like the antigens that attack the human being. To defend against these antigens, the human immune system performs a negative selection process to separate self substances from nonself ones. The result of this process is a set of receptors that identify the antigens. This negative selection process served as a basis for our signature generation process that store the relevant malware characteristics. These signatures work as receptors to identify malicious code.
- To identify an antigen, the cells circulate in the human body searching for any foreign substance. This corresponds to the network and the behavior agents that analyze data captured on a monitored computer system in order to detect malware.
- If the immune system detects any antigen, an immune response is triggered to perform the identification and removal of the antigen. The same happens in our proposed system. The network and behavior agents generate a detection report containing information about the identified malware and how to remove it.
- Finally, the human immune system has learning mechanisms and memory (knowledge) that help in identifying future occurrences of such antigens. This feature is performed by the signatures database in the proposed system. This database enables the detection of new attacks of malware.

Table 3. Analogy between the human immune system and our proposed system

Human Immune System	Malware Detector System proposed
Antigens	Malware
Negative selection	Signature generation
Cell receptors that identify an antigen	Behavior and Network signature
Cell that identify an antigen	Behavior and Network agent
Immune response	Detection report
Learning and memory mechanisms	Signature Database

We performed tests using a monitored environment to validate our approach and the results were promising, indicating that both the behavior and the network detector are effective in detecting a malware attack, since alerts were issued for malware samples whose behavior pattern matched a signature stored in our database. Limitations of our work includes the analysis platform, that is currently tied to Windows XP operating systems and the need to periodically redo the training step to update the decision tree e the signature database. As a future work, we intend to generalize the detection process to encompass malware targeting other platforms.

References

1. Bin, L., et al.: A NetFlow based flow analysis and monitoring system in enterprise networks. *Computer Networks* 52, 1074–1092 (2008)
2. Cert. Cartilha de Segurança para a Internet - Parte VIII: Códigos Maliciosos (Malware) (2006), <http://cartilha.cert.br/download/cartilha-08-malware.pdf>
3. Cisco. Introduction to Cisco IOS NetFlow - A Technical Overview (2007), http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/prod_white_paper0900aecd80406232.pdf
4. Claise, B.: RFC 3954: Cisco Systems NetFlow Services Export Version 9 (2004), <http://www.ietf.org/rfc/rfc3954.txt>
5. Corrêa, J.L., Proto, A., Cansian, A.M.: Modelo de armazenamento de fluxos de rede para análises de tráfego e de segurança. VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg) 8, 73–86 (2008)
6. Dasgupta, D., Niño, L.F.: *Immunological Computation Theory and Applications*, vol. 1, p. 296. Taylor and Francis Group (2008)
7. Dionaea. Dionaea — catches bugs (2011), <http://dionaea.carnivore.it/>
8. Filho, D.S.F., et al.: Análise Comportamental de Código Malicioso através da Monitoração de Chamadas de Sistema e Tráfego de Rede. X Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg) 10, 202–212 (2010)
9. Forrest, S., et al.: Self-nonsel Self-Discrimination in a Computer. In: *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, vol. 10, pp. 311–324 (1994)
10. Grégio, A.R.A.: Aplicação de técnicas de Data Minings para a análise de logs de tráfego TCP/IP. Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada (2007)

11. Han, J., Kamber, M.: Data Mining Concepts and Techniques, vol. 3, p. 550. Morgan Kaufmann Publishers (2006)
12. He, Y., Yiwen, L., et al.: A Model of Collaborative Artificial Immune System. In: 2nd International Asia Conference on Informatics in Control, Automation and Robotics, vol. 3, pp. 101–104 (2010)
13. Hsiao, H.W., Chen, D.N., Wu, T.J.: Detecting Hiding Malicious Website Using Network Traffic Mining Approach. In: 2nd International Conference on Education Technology and Computer (ICETC), vol. 5, pp. 276–280 (2010)
14. Kephart, J.O.: A Biologically Inspired Immune System for Computers. Artificial Life IV, 130–139 (1994)
15. Paula, F.S.: Uma arquitetura de segurança computacional inspirada no sistema imunológico. Tese de Doutorado (Doutor em Ciência da Computação) UNICAMP Instituto de Computação (2004)
16. Stalmans, E., Irwin, B.: A Framework for DNS Based Detection and Mitigation of Malware Infections on a Network. In: Information Security South Africa (ISSA), pp. 1–8 (2011)
17. Virus Total. About VirusTotal (2011), <http://www.virustotal.com/about.html>
18. Witten, I.H., Frank, E.: Data mining: practical machine learning tools and techniques with java implementations, vol. 3, p. 629. Morgan Kaufmann Publishers (2000)

Interactive, Visual-Aided Tools to Analyze Malware Behavior

André Ricardo Abed Grégio^{1,2}, Alexandre Or Cansian Baruque², Vitor Monte Afonso², Dario Simões Fernandes Filho², Paulo Lício de Geus², Mario Jino², and Rafael Duarte Coelho dos Santos³

¹ Renato Archer IT Research Center (CTI/MCT), Campinas, SP, Brazil
argregio@cti.gov.br

² University of Campinas (Unicamp), Campinas, SP, Brazil
{vitor,dario,paulo}@las.ic.unicamp.br, jino@dca.fee.unicamp.br,
orcansian@gmail.com

³ Brazilian Institute for Space Research (INPE/MCT), S. J. dos Campos, SP, Brazil
rafael.santos@lac.inpe.br

Abstract. Malicious software attacks can disrupt information systems, violating security principles of availability, confidentiality and integrity. Attackers use malware to gain control, steal data, keep access and cover traces left on the compromised systems. The dynamic analysis of malware is useful to obtain an execution trace that can be used to assess the extent of an attack, to do incident response and to point to adequate counter-measures. An analysis of the captured malware can provide analysts with information about its behavior, allowing them to review the malicious actions performed during its execution on the target. The behavioral data gathered during the analysis consists of filesystem and network activity traces; a security analyst would have a hard time sieving through a maze of textual event data in search of relevant information. We present a behavioral event visualization framework that allows for an easier realization of the malicious chain of events and for quickly spotting interesting actions performed during a security compromise. Also, we analyzed more than 400 malware samples from different families and showed that they can be classified based on their visual signature. Finally, we distribute one of our tools to be freely used by the community.

Keywords: Security data visualization, malware analysis.

1 Introduction

Malicious software—malware—is the main current threat to information systems security. It is usually spread through the Internet and can cause incidents with severe damage to confidentiality, integrity or availability of systems and data. Most malware are targetless, attacking as many systems as they can, so that an attacker can gain control and use of the victim's resources or steal sensitive data. However, there are cases in which malware have specific targets and are thoroughly designed to delude the victim, talking the unsuspecting user into

supplying confidential information, as it happens in attacks directed to a government infrastructure. In both situations, severe incidents caused by malware can disrupt an entire network of systems by disseminating through headquarters and then to branch offices or can also cause irreparable damage by exposing confidential information.

When attacks succeed in breaking into a computer system, forensic procedures can be followed in order to find out:

- How the attack was perpetrated;
- Where the points of collection of information are (downloading of tools and sending of data);
- What happened during the attack to the system.

In the case of malware attacks, it is important to collect the binary that infected the system or was downloaded after the target system was compromised. This binary may then provide some clues leading to a deeper understanding of the techniques used by the attacker and the purpose of the attack. This can be done by running this malware in a controlled environment and monitoring all the filesystem and network activities to compose a specific behavioral trace.

Malicious behavioral traces are in essence a log of the events performed by a malware on a compromised system, but this amounts to large chunks of data. Such logs are difficult to analyze as we have to stress out interesting segments of behavior (main malicious actions) while simultaneously having to obtain a general overview of the extension of the damage. However, the information obtained from this analysis is paramount to provide adequate incident response and mitigation procedures. In those cases of massive amounts of textual data to analyze, we can apply visualization techniques that can greatly enhance the analysis of logs and allow us to quickly spot important actions performed by a specific malware and to better understand the chain of malicious events that led to the compromise of the target system.

The main contributions of this article are:

- We developed two visualization tools—the behavioral spiral and the malicious timeline—to aid security analysts to observe the behavior that a malicious software presents during an attack. Those tools are interactive and they allow a user to walk through the behavior while performing zoom, rotate, and gathering of detailed information for each malware action.
- We discuss visual classification of malware families and show that our tool can be used to visually identify an unknown malware sample based on its comparison to previously known malware, and that is a step towards a visual dictionary of malicious code.
- We distribute online a beta version of our prototype, so that the community can benefit from it and use it freely.

2 Related Work

There are several research works that use visualization tools to overcome the plenty of data provided by textual logs related to security. Some of them are not

open to the public, others are neither intuitive to use nor interactive, and there are those whose results are more difficult to be visually interpreted by security analysts than if they search manually through the log files.

Quist and Liebrock [12] applied visualization techniques to understand the behavior of compiled executables. Their VERA (Visualization of Executables for Reversing and Analysis) framework helps the analysts to have a better understanding of the execution flow of the executable, making the reverse engineering process faster.

Conti et al. [3] developed a system that helps the context-independent analysis of binary and data files, providing a quick view of the big picture context and internal structure of files through visualization. In a forensic context it is especially helpful when analyzing files with undocumented formats and searching for hidden text messages in binary files.

Trinius et al. [15] used visualization to enhance the comprehension of malicious software behavior. They used treemaps and thread graphs to display the actions of the executable and to help the analyst identify and classify malicious behavior. While their thread graphs can confuse a human analyst with lots of overlapped information and lack of interactivity, our timeline (Section 4.1) allows a walk-through over the chain of events performed by different processes created and related to the execution of a certain malware sample, as well as the magnification of interesting regions, information gathering about selected actions and annotation. Furthermore, our behavioral spiral represents temporal action, whereas their proposed treemaps consist of the actions' distribution frequency. Again, there is a lack of interactivity and excessive data, as we handle only actions that can cause changes on the target system. However, similarly to our work, it is not possible to visually classify every malware family, as variant samples can pertain to a class while presenting completely different behavior regarding the order or nature of the performed actions.

Finally, reviewing logs from intrusion detection systems is an important task to identify network attacks and understand these attacks after they happened. There are several tools that use visualization for this purpose; each one has its own approach and is better than others at specific situations. To take advantage of those tools the DEViSE (Data Exchange for Visualizing Security Events) framework [13] provides the analysts a way to pass data through different tools, obtaining a better understanding of the data by aggregating more extracted information.

3 Data Gathering

To visualize malware behavioral data, we first need to collect malware samples that have been currently seen in the wild and then analyze them to extract the actions they would perform in an attack to a target system. In the sections below, we discuss our approaches to malware collection and behavior extraction.

3.1 Malware Collection

To collect malware samples that spread through the Internet, we use the architecture described in [5], which uses mixed honeypots technology (low and medium interaction) [10] to capture malicious binaries for MS Windows systems. Honeypots are systems that are deployed to be compromised so as to lure attackers to reveal their methods and tools by the compromise of a highly controlled environment. The collection architecture consists of a Honeyd [11] node to forward attacks against certain vulnerable ports to a Dionaea [7] system, which emulates vulnerable MS Windows services in those forwarded ports and actually downloads the malware sample. During 2010 we captured more than 400 unique samples, which are used as our test dataset in this article.

3.2 Behavior Extraction

To extract the behavior of a malicious software, we run it in a controlled environment and monitor the actions it performs during the execution in the target system. Those actions are based on the system calls executed that are relevant to security, i.e. if they modify the status of the system or access sensitive information, such as file writing, process creation, changes in registry values or keys, network connections, mutex creation and so on. The dynamic analysis environment used for behavior extraction is BehEMOT [6], a system that produces logs in which each line means one action performed by the monitored malware sample or a child process and is in the form “*timestamp, source, operation, type, target*”. For instance, let's suppose that a malware sample “*mw.exe*” created a process entitled “*downloader.exe*”, which connects to port 80 of an Internet IP address *X.Y.W.Z* to download a file *a.jpg* to a temporary location *TEMP*. The log file produced by BehEMOT would have the following three lines:

```
ts1, mw.exe, CREATE, PROCESS, downloader.exe
ts2, downloader.exe, CONNECT, NET, X.Y.Z.W:80
ts3, downloader.exe, WRITE, FILE, TEMP/a.jpg
```

Therefore, we use the BehEMOT behavior format to input the textual data to our visualization tools, which are described in the next section. It is worth noting that logs produced from malware sample execution can be thousands of lines long, motivating the use of visual tools to aid the process of human analysis.

4 Interactive Visualization Tools for Behavioral Analysis

The behavior of a malicious program can be interpreted as a chain of sequential actions performed on a system (as seen in the previous section), which involves operating system interactions and network connections. The analysis of those operations can provide the steps that were performed in an attack to understand the incident as a whole, as well as detailed information about what was changed

in a system, such as libraries that were overwritten, infected files, downloaded data and even evaded information.

Usually, antivirus developers analyze unknown malware samples to create signatures or heuristics for detection. This is an overwhelming process and involves plenty of manual work, as a human analyst has to search for pieces of data that characterize the sample as part of an already known malware class or create a new identifier to it. Actually, this process is worse due to the increasing amount of new malware made from ‘do-it-yourself’ kits and variants of older ones. Sometimes, a malware is assigned to a family (and detected by an antivirus engine) based on the value of a mutex it creates, or on a specific process that it launches with a particular name, or on the kind of information it sends to the network.

Our motivation to develop visual tools is to make it easy to process and pinpoint very specialized information and then to help a human analyst to focus in the interesting actions performed by a malware sample. This way, it is possible to quickly analyze new malware by visualizing their overall behavior and expanding only those actions that an experienced analyst considers suspicious or important. Also, public available dynamic analysis systems (e.g. [8], [14], [1]) provide textual reports full of information that would be easier to be interpreted if an analyst could visually manipulate it. To fill this gap, we developed two tools that transform a textual report from a dynamic analysis system into an interactive and visual behavior, which are explained below.

4.1 Timeline and Magnifier

Malware time series events can also be visualized in simple x-y plots, where the x axis represents the time and the y axis some information about the event. The time information on the x axis can be any of the following: i) the absolute time of occurrence of an event; ii) the relative time of occurrence (counted from a particular initial value); or iii) a simple sequence that implies the order of occurrence of the events.

The event information can be plotted using several different methods, often specifically tailored to a particular purpose. The height on the y axis can be used to represent the frequency of occurrence, severity or intensity of a given event, if such information is known, or a discrete representation of event types. Decorations such as different marks for additional event characteristics can also be used to allow representation of more data dimensions than just two. Additional graphical elements may also be used, but one should always take care not to overload the plot with too much graphical content, which may confuse the user and hinder his/her ability to draw quick conclusions from the plot.

An example x-y plot used to represent malware time series events is given in Figures 1 and 2. They show the corresponding tool in action, as it parses a malicious behavior file with malware events ordered by action timestamp and plots the result using a simple, interactive interface. The tool draws the whole time series in two panels: on the top panel all points on the series are plotted, with the x axis representing the order in which events occurred and the y axis

representing the action associated with an event (which are not in any particular order and can be rearranged). Also, events are plotted as dots of different colors, according to the process id that performed such actions. For example, if the malware associated process created two child processes and also required service from an already running process, we would have four different colors in the graphic: malware, first child, second child and the running process). Not all possible monitored actions have to be performed by a malware, so the y axis varies accordingly to what was present in the captured behavioral trace.

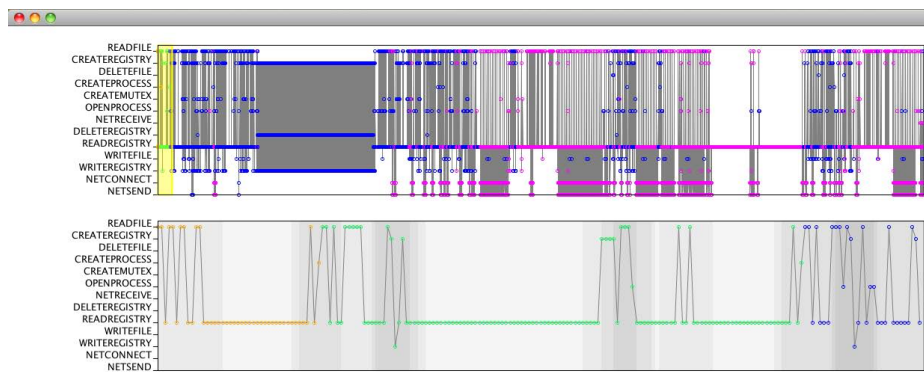


Fig. 1. Timeline and Magnifier tool representing malicious events

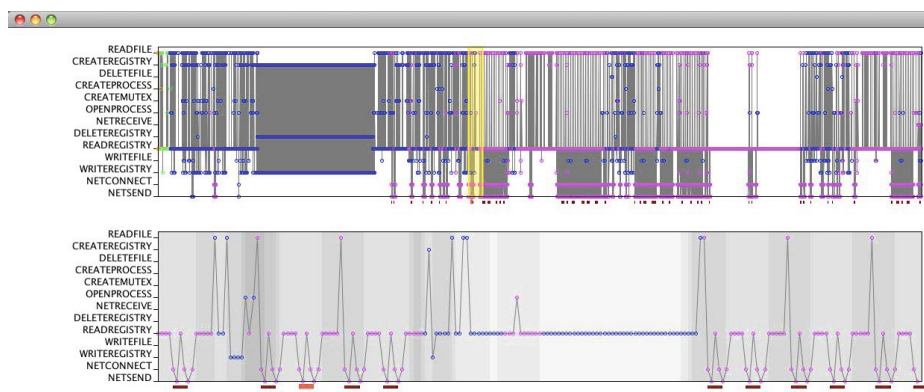


Fig. 2. Sequence of events selected by the user as a pattern to be searched (light red) and automatically matched events (dark red)

Plots created by this tool are also interactive. Since there are many events on the top section of the plot, it is hard to see exactly which one follows which one, so a region of interest (highlighted under a translucent yellow region on

the plot) can be selected by the user. Selection is done by dragging the region with the mouse, which also causes the region of interest to be shown enlarged on the bottom panel of the plot. The x and y axes and plot colors follows the ones in the top section. The bottom part of the plot also conveys information about the diversity and variability of operation types in its gray background: darker backgrounds suggest a higher diversity, whereas lighter backgrounds point to higher similarity.

4.2 Malicious Spiral

The goal of this tool is to present an ordered sequence of all malicious actions of an attack in a spiral format, using an iconic representation. The spiral representation is useful to show the big picture of a malware sample behavior and also to allow quick visual comparisons among different malware samples even in the presence of variants. Instead of using a straight line broken in columns (as seen in [4]), the spiral format is less prone to confusion as small variances present in the behavior of malware from the same class usually keep the general visual appearance that models a family.


















By exploiting the viewing abilities available, an investigator can zoom in and out, turn, tilt, select behavior slices, view the logged action in textual form and compare it with other behavioral data, if available. Thus, we provide not only an overview of the attack, but also the possibility of identifying certain behavioral patterns that could help in the classification of malware samples (Section 5).

The operations and types that are monitored and used to produce a behavior log are shown in Table 1, as well as all icons that are used to represent them. We divided the table lines in four groups of operations with similar purpose on each subsystem type, e.g., a CONNECT operation in a NET type has the same effect as that of a CREATE REGISTRY, i.e they prepare the environment for an active operation that will represent a registry value being written or a network connection being opened that later might send data. In the same way, a READ FILE or REGISTRY, a RECEIVE NET and a QUERY MUTEX are all passive operations, and so on.

5 Tests and Analysis of Results

Although we have developed the timeline and the spiral tools based on the same kind of log format, they have different usage. The timeline and magnifier tool can be used by any user to do a “behavioral walk-through”, while verifying how many processes were launched, what actions they performed and from what kind, etc. On the other hand, the spiral tool can be used to visually identify malware from the same family, while simultaneously depicting an iconic overview of the behavior that can be manipulated to show more detailed information (present in the log file).

Table 1. Monitored operations and types grouped by activity equivalence and differentiated by icons and colors

Action / Type	MUTEX	FILE	PROC	REG	NET
READ					
QUERY					
RECEIVE					
WRITE					
SEND					
CONNECT					
CREATE					
DISCONNECT					
DELETE					
TERMINATE					
RELEASE					

In this section, we show how the spiral tool serves as a visual dictionary and how it can help classify malware from the same family. The current prototype can be obtained from [9], together with larger screenshots. To the extent of our tests, we extracted the execution behavior from 425 malware samples collected by the system described in Section 3.1 and dynamically analyzed them with the system described in Section 3.2.

All malware samples from our dataset are currently found “in the wild” and together constitute variants from 31 families. Scanning them with the up to date ClamAV antivirus engine [2] reveals 94 unidentified samples. In [9] one can also verify the behavioral spiral pictures for each analyzed malware and its respective log.

By observing the generated spirals, we realized that it is possible to group some of them by their visual behavior. Also, malware samples from different families present similar behavioral patterns among samples from their own class, while at the same time keeping a dissimilarity from other classes’ samples. This differentiation factor present in the visual patterns is an important indicative that clustering algorithms, artificial intelligence and data mining techniques can be applied to our logs to classify malware based on behavioral similarities.

In Figure 3, we chose two trojan families—Pincav and Zbot—and selected three samples of each one to be printed side-by-side. Notice that even when the

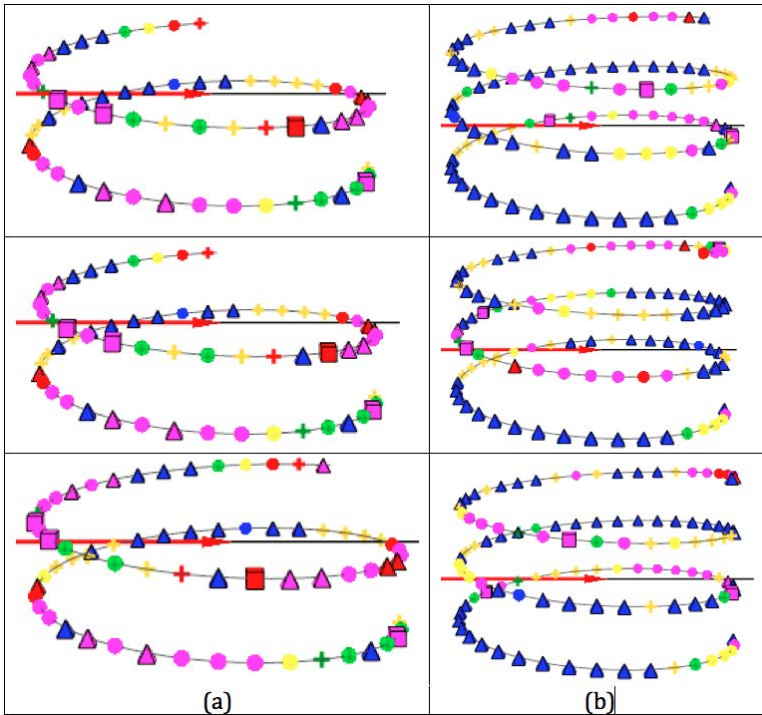


Fig. 3. Behavioral spirals from three samples of the trojan families ‘Pincav’ (a) and ‘Zbot’ (b)

malware samples from a family perform variant operations, they still keep a behavioral pattern that can be used to characterize them in the same class.

Another interesting fact is that if a malware sample tried to do more network connections than another one from the same family (e.g., malicious scans) or if it performed fewer actions or crashed, it is possible to clearly notice the similarities between an incomplete behavior and a larger one, as shown in Figure 4, which contains three samples of the *Allapple* family.

In Figure 5, we depict the behavioral spirals from four different malware families—the worms Palevo and Autorun; the trojans Buzus and FakeSSH. Notice that we can visualize the separability of the classes with minor variances among behaviors from the same family for Palevo, Autorun and FakeSSH. In the case of Buzus, one can observe that the first two behaviors significantly differ from the last ones, putting those samples apart from each other is an automated classification scheme. However, in Figure 6, one can also realize that a sample whose AV assigned label is “UNKNOWN”—i.e. an unidentified sample—presents a visual behavior that is quite similar to a sample from the trojan family “Inject”.

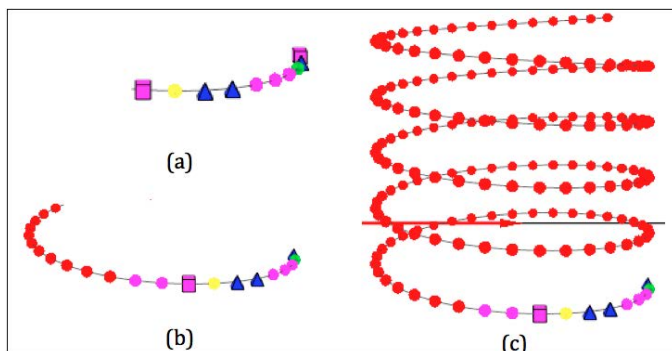


Fig. 4. Three ‘Allapple’ worm variants showing: (a) a sample that could not connect to the network and stopped its activities; (b) a sample that performed a short network scan; (c) a sample performing a massive scan, where each red sphere means a connection to a different IP address

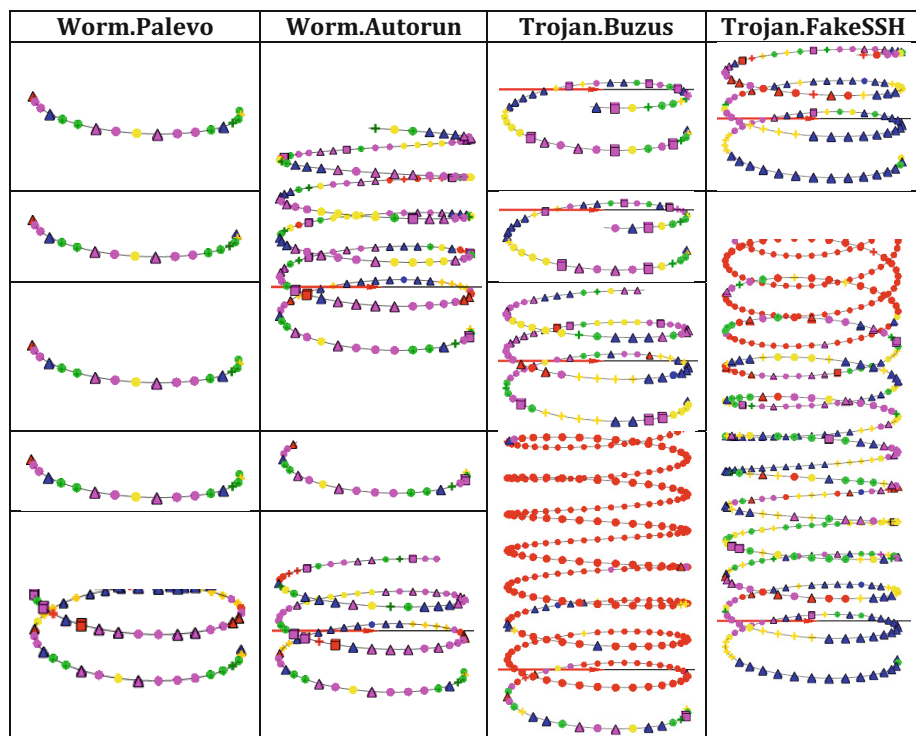


Fig. 5. Visual behavior extracted from four malware families

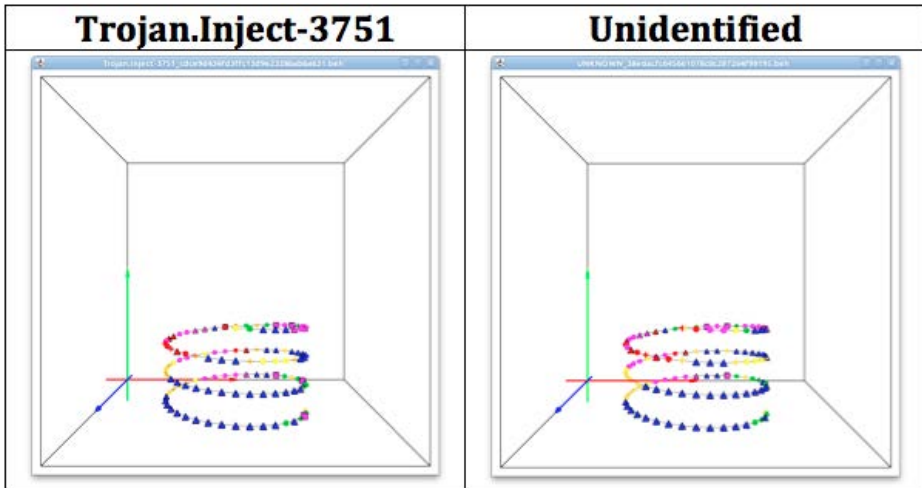


Fig. 6. Unidentified malware (right) sample visually classified as a known threat—Inject trojan (left)

6 Conclusion and Future Work

In this article we propose two interactive, visual-aided tools to increase the efficiency in malware analysis, which allow an overview of malicious behaviors to security analysts. Moreover, our tools allow a walkthrough over the logs, the annotation and emphasis on interesting actions, the searching for patterns, a deep understanding of the damage performed on a target system and the visual comparison among malware samples. Hence, the possibility of visual family differentiation indicates that we can apply, in the future, an automated technique to classify, to cluster or to mine behavioral data. Also, it is possible to visualize which parts of a malware sample behavior are similar to another one's, indicating the same functionality or even code reuse. We are developing, as future works, a behavioral database that can bring some intelligence to the annotation process of the timeline/magnifier tool, the ability to load multiple logs at the same time to visualize several spirals in parallel and, finally, a classification algorithm to be integrated to the spiral tool that will allow us to automatically identify samples that share a high level of similarity and, after that, visualize their behavior.

References

1. Buehlmann, S., Liebchen, C.: Joebox: a secure sandbox application for windows to analyse the behaviour of malware, <http://www.joebox.org>
2. Clam antivirus, <http://www.clamav.net>
3. Conti, G., Dean, E., Sinda, M., Sangster, B.: Visual Reverse Engineering of Binary and Data Files. In: Goodall, J.R., Conti, G., Ma, K.-L. (eds.) VizSec 2008. LNCS, vol. 5210, pp. 1–17. Springer, Heidelberg (2008)

4. Eick, S.G., Steffen, J.L., Sumner Jr., E.E.: Seesoft—A Tool for Visualizing Line Oriented Software Statistics. *IEEE Transactions on Software Engineering* 18(11), 957–968 (1992)
5. Grégio, A.R.A., Oliveira, I.L., dos Santos, R.D.C., Cansian, A.M., de Geus, P.L.: Malware distributed collection and pre-classification system using honeypot technology. In: *Proceedings of SPIE*, vol. 7344, pp. 73440B–73440B-10 (2009)
6. Grégio, A.R.A., Fernandes Filho, D.S., Afonso, V.M., dos Santos, R.D.C., Jino, M., de Geus, P.L.: Behavioral analysis of malicious code through network traffic and system call monitoring. In: *Proceedings of SPIE*, vol. 8059, pp. 80590O–80590O-10 (2011)
7. The HoneyNet Project. Dionaea, <http://dionaea.carnivore.it>
8. Kruegel, C., Kirda, E., Bayer, U.: Ttanalyze: A tool for analyzing malware. In: *Proceedings of the 15th European Institute for Computer Antivirus Research (EICAR 2006) Annual Conference* (2006)
9. MBS Tool. Malicious Behavior's Spiral - Beta version, <http://www.las.ic.unicamp.br/~gregio/mbs>
10. Provos, N., Holz, T.: *Virtual Honeypots: from botnet tracking to intrusion detection*. Addison-Wesley Professional (2007)
11. Provos, N.: Honeyd - A Virtual Honeypot Daemon. In: *10th DFNCERT Workshop* (2003)
12. Quist, D., Liebrock, L.: Visualizing Compiled Executables for Malware Analysis. In: *Proceedings of the Workshop on Visualization for Cyber Security*, pp. 27–32 (2009)
13. Read, H., Xynos, K., Blyth, A.: Presenting DEViSE: Data Exchange for Visualizing Security Events. *IEEE Computer Graphics and Applications* 29, 6–11 (2009)
14. ThreatExpert, <http://www.threatexpert.com>
15. Trinius, P., Holz, T., Gobel, J., Freiling, F.C.: Visual analysis of malware behavior using treemaps and thread graphs. In: *International Workshop on Visualization for Cyber Security (VizSec)*, pp. 33–38 (2009)

Interactive Analysis of Computer Scenarios through Parallel Coordinates Graphics

Gabriel D. Cavalcante¹, Sebastien Tricaud²,
Cleber P. Souza¹, and Paulo Lício de Geus¹

¹ Institute of Computing, University of Campinas
Albert Einstein, 1251, 13083-852 – Campinas – Brazil

² Picviz Labs
40 Avenue Guy de Collongue, 69130 – Ecully – France

Abstract. A security analyst plays a key role in tackling unusual incidents, which is an extenuating task to be properly done, a single service can generate a massive amount of log data in a single day. The analysis of such data is a challenge. Among several available techniques, parallel coordinates have been widely used for visualization of high-dimensional datasets and are also highly suited to plot graphs with a huge number of data points. Unusual conditions and rare events may be revealed in parallel coordinates graph when they are interactively visualized, which is a good feature for the analyst to count on. To address that, we developed the Picviz-GUI tool, adding interactivity to the visualization of parallel coordinates graph. With Picviz-GUI one can shape a graph to reduce visual clutter and to help finding patterns. With a set of simple actions, such as filtering, changing line thickness and color, and selections, the user can highlight the desired information, search through the variables for that subtle data correlation. Picviz-GUI visualization helps the security analyst to understand complex and innovative attacks, to later tune automatized classification systems. This article shows how features on top of parallel coordinates graph can be effective to uncover complex security issues.

1 Introduction

Nowadays, a security administrator is responsible for the analysis of large amounts of data that represent activities on the network as a whole. System logs and network traffic can provide useful information to describe what is going on with the infra-structure. However, a single service can produce a myriad of lines of log a day. Combined with the complicated topologies of private networks, this really might constitute a overwhelming volume of data.

Normally, computer evidence creation is instructed by its operator through service configuration files. Even so, some useful information might be hidden from inspection, requiring special tools and techniques for the analyst to become aware of them [6].

A large range of software tools aid investigators in finding illegal activities on affected systems. These tools reduce tedious efforts on the part of the examiner, especially when searching for some weird event in a very large amount of data. In addition, the investigators need correlate and interpret such data, which by itself is an error-prone

process that consumes an abundance of time and patience. This is especially true in the absence of a hint or particular reason to suspect anything.

Visualization techniques can help forensic specialists direct their efforts to suspicious events, processes, or hosts, through assisting the data interpretation process. Furthermore, one can draw from the visualization world how to approach the problem.

Since parallel coordinates graph can represent large datasets in multiple dimensions, handling their multivariate abilities help factor out different categories of computer security data, whenever this is needed.

Picviz [12] provides mechanisms to automate the static graph creation in a simple way, but maybe user cannot get all suspicious events only with static images and filters. In this paper we present Picviz-GUI, that provides a powerful interactive interface to parallel coordinates.

2 Brief Explanation about Parallel Coordinates

Parallel-coordinates—hereafter called *||-coordinates*— is a famous visualization technique which plots individual data across many dimensions. This is feasible because any individual data element can be described as a tuple.

Let us imagine how much different information one can grab about a type of data, for example, a soccer league table (position, team name, number of matches, wins, draws, losses, goals conceded, goals scored, number of points, etc.). To complete a table like that we must have many tuples containing a value for each of these data elements.

In a formal way, we could define each line of the soccer table as a vector v described as a tuple $(x_1, x_2, x_3, \dots, x_n)$ which could be represented on an N -dimensional plane \mathbb{R}^N . Unfortunately, when n becomes bigger than 3 it is difficult to plot n -dimensional vectors using a 3-dimensional physical space.

||-coordinates overcome the 3-dimensional space limitation by adding a parallel axis to each dimension of v . As such, the vector $(x_1, x_2, x_3, \dots, x_n)$ is visualized by plotting x_1 on axis 1, x_2 on axis 2, and so on through x_n on axis n . In this way, each $v \in \mathbb{R}^N$ is represented by a polygonal line \vec{V} .

As goes the old say, a picture is worth a thousand words, so one had better see an image than read information. Figure 1 shows a trivial representation of *||-coordinates* to a single extract of network activity. It uses a set of parallel lines (not necessarily vertical), where each one corresponds to an axis of data from the table.

Each row of the table is then represented by a polygonal line across these axes, such that the point at which it crosses each axis corresponds to the value of the tuple's data point on that axis. The data plotted on this example is borrowed from Section 3.2. For additional understanding and information we highly recommend literature such as [4] and [16].

3 Creation of Parallel Coordinates

||-coordinates are commonly used for visualizing multivariate data. Recent work has evaluated how efficient it can be for different tasks, in comparison with other multidimensional techniques [8][3].

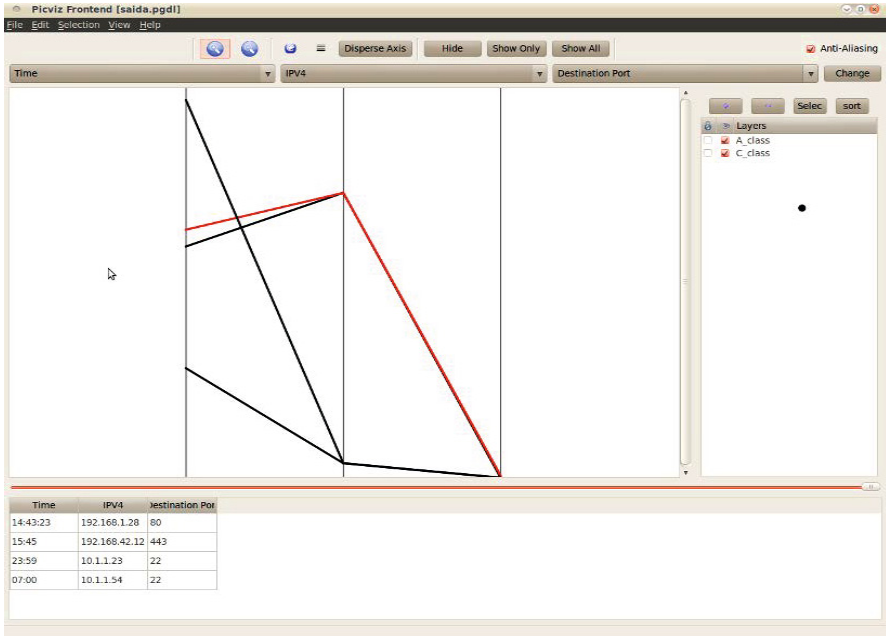


Fig. 1. Picviz-GUI showing the example described in Section 3.2 with layers

Picviz [12] is a suite of tools that automate the creation of *ll-coordinates* images, either from logs or any other data representation, though we are mainly concerned with data from log files here.

The Picviz suite contains a well-defined API that can be used to transform information into *ll-coordinates* points, acting like a library, so that different applications can be built on top of it to take advantage of its mapping process.

As described in [12], the Picviz suite has a Command Line Interface (CLI) that allows the creation of static *ll-coordinates* graphs from thousands of lines of information in a few seconds. The same tool allows a user to filter some characteristics and use some output plugins like frequency analysis.

Despite the fact that the CLI allows handling a huge number of lines, it forces the analyst to follow a sequence of steps: adjust filters and plugins; run the application and then find results in the image output. After many repetitions, this process may become tedious. In addition, it might be difficult to filter, using generic expressions, a large number of specific events.

In reality, unusual conditions and rare events may also be revealed with the nature of *ll-coordinates* graphs when they are interactively visualized, so in that way the user can filter and apply different attributes to the graph and have on-the-fly response. Picviz-GUI was built to address this issue.

3.1 Data Transformation

To transform data into a *||-coordinates* plot image, the user first needs access to the data that will be plotted.

From the security analysis viewpoint, one needs to analyze the largest possible amount of data that can describe something about the whole scenario.

By following the lead provided by a user's complaint, for instance, the analyst can focus on specific data that might be related to suspicious events. For example, if a web browser has an abnormal behavior, related events might be found on the web browser log.

Typically, important data result in large amounts of information, and to deal with it we can use a process called **parsing**. Parsing, the process of identifying individual tokens of information, takes here an extra meaning, since data coming from different sources may be combined to return a single output. The parsing process in the Picviz workflow should transform captured logs into the Picviz Graph Description Language (PGDL).

3.2 Picviz Description Language – PGDL

From logs to the graph, the PGDL language aids in structuring the information, allowing users to define how data will be treated on graph creation through three main sections: header, data and axes.

The header section defines the title of the graph, whereas the axes define how data will be represented on the graph, describing each axis of the *||-coordinates* graph and their type.

The data section contains all events that will be represented in the graph; each line must comply with the format specified in the axes section. In PGDL, each dimension is represented by an axis, which is defined in the axes section:

```
header {
    title = "See the different groups of IPs";
}
axes {
    timeline t [label = "Time"];
    ipv4      i;
    integer   dport [label = "Destination Port"];
}
```

The keyword before each axis name defines the way data shall be placed on the axis. It is similar to variables in programming languages: integer, short, string, timeline, enum etc.

The types described in the axes section for each variable are tips used by the Picviz engine for the graph construction process; for example, the engine will predict 2^{32} possible positions for the integer axis in the example above. Once the axes section is defined, data must be inserted into the data section as shown below:

```

data {
  layer C_class {
    t="14:43:23", i="192.168.1.28", dport="80";
    t="15:45", i="192.168.42.12", dport="443" [color="#FF0000"];
  }
  layer A_class {
    t="23:59", i="10.1.1.23", dport="22";
    t="07:00", i="10.1.1.54", dport="22";
  }
}

```

“Data” is the original data from the log file, with dimensions being comma-separated. If needed, the subsection “layer” could be added to the data section to describe layers. It is a good way to ease the search for relevant information.

3.3 Picviz-GUI

As mentioned in the introduction, the *||-coordinates* graph really shows its potential when it is used interactively. An interactive frontend was written for this purpose (see Fig. 1). Picviz-GUI is responsible for the creation and representation of *||-coordinates*. It was written in Python and Trolltech’s QT4 library to provide a skillful interaction toward finding relations among variables, allowing one to apply filters, drag the mouse over the lines to see information displayed, change axis order on the fly, zoom, brush lines and change line thickness.

One of *||-coordinates* disadvantages is the choice of a suitable order for the axes intended to display relationships among variables. There has been a variety of proposed automatic solutions to select the best sequence of axes in *||-coordinates*. However, greater satisfaction comes with user manipulation of the axes order [17]. The top bar allows the user to reorder axes on the fly, which provides for an effortless search for the best representation for the data under analysis; this is done without any modifications to the parser. All of the Picviz Suite tools are open-source under **GPLv3** and can be obtained from [18].

4 SSH Authentication Log Analysis

This section shows many operations under Picviz-GUI that conduces analysis through an interactive ambient. The analyst perceives your actions on-the-fly and is able to create layers to isolate events to focus on these subsets. SSH (*Secure Shell*) is widely used to provide remote access to a Unix-like host. Its log files keep track of successful and failed login attempts, time, status and source IP address [9].

As will be showed above, *||-coordinates* have inherent property that massive attacks can have a unique graphical pattern that enables the network administrator to rapidly detect and respond. But the major asset of Picviz-GUI is your interactive property, so

¹ A professional version of Picviz suited to handle even larger amounts of events can be obtained from Picviz Labs website.

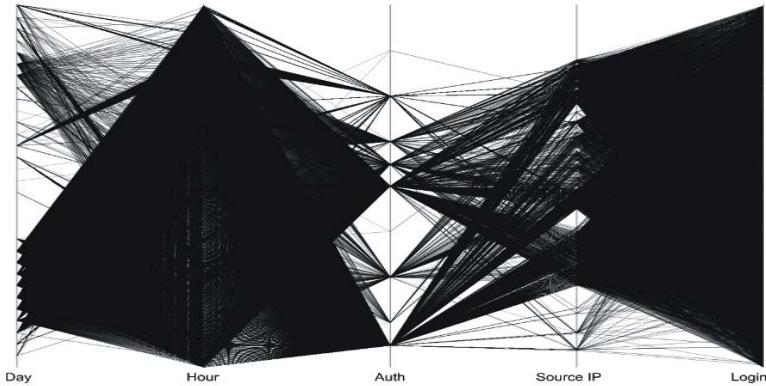


Fig. 2. Plot of raw data without any analysis

playing with the graph the administrator can understand why your defense systems maybe are ineffective.

Each log analysis requires from the analyst the knowledge about what is provided by the logging mechanism for that service and how to tune it if necessary. A traditional log analysis scarcely allow the detection of anomalies such as distributed brute-force scan attempts, because critical data are sparsely distributed over long log files encompassing many days, which helps attack activities to pass unnoticed.

Figure 2 plots the day, hour, auth (authentication status), source IP and login axes for each entry of an entire month of a SSH server log file. The careful selection of the right axis and the data plot order depends on the information that needs to be retrieved and how the analyst wants to view the information. Layers and coloring features help group suspicious data and provide him/her with a range of possibilities to look into the data for intelligence gathering.

The first step after plotting raw data is to try and detect concentration areas in order to isolate events in layers. At the bottom of Day axis, the most active days could be noted.

Another good point to start is the Auth axis, which describe the status about each remote login attempt. These may indicate abuses in that service as noticed on the auth and login axes in Figure 2, related primarily to authentication denied status for "Invalid user" and "User not allowed" attempts. By isolating these candidates it was possible to confirm the existence of two types of massive login attempts.

By simply picking a color for each of most active days, the first type is clearly identified: an individual source IP address tries many combinations of usernames and passwords in a short time window, as detached in Figure 3, an attack that was probably performed by some existing automated tool.

This method is not very effective from the attacker's point of view because traditional defense tools are prepared to implement deny rules for these cases, since the high activity performed by a single IP source address [11] is easily identified by simple heuristics. On this point of analysis, only authentication errors were selected to be plotted in this excerpt.

As for the second type, many source IP addresses try combinations of usernames and passwords, but do so by implementing special evasion techniques.

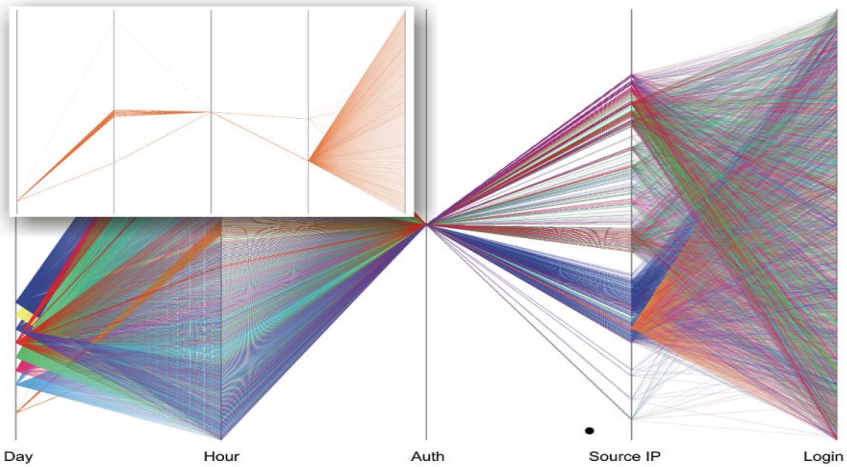


Fig. 3. Graphic showing a layered version of the data for authentication errors by unknown logins. Highlighted is a detected single source IP address SSH brute-force attack.

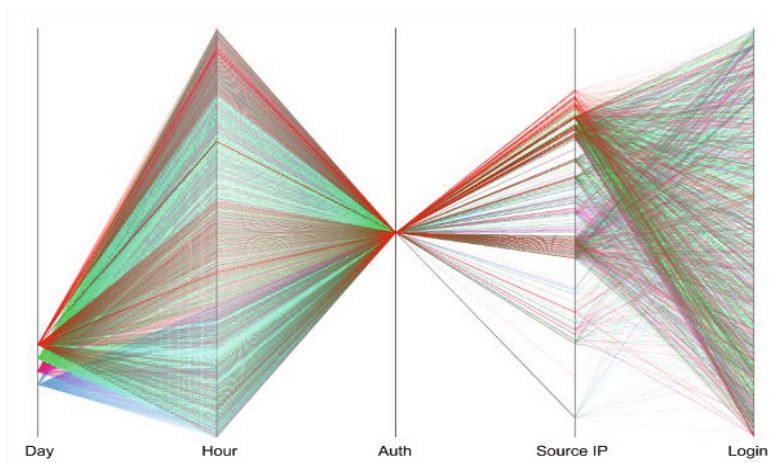


Fig. 4. The most active days, initial 4-day period

The suspect is confirmed with details in Figure 4 where a good concentration of red (day 7) events could be observed. The red day have a massive amount of login attempts distributed over different ips, a closer look in the top half of Hour axis reveals a “coordinated stop” of red login attempts.

This property allows the conclusion that a third element on the network was coordinating the login attempts by a variety of sources, a behavior that characterizes the usage of botnets².

² A group of compromised machines that keep their normal functionality for valid users but are simultaneously serving a remotely commanding attacker.

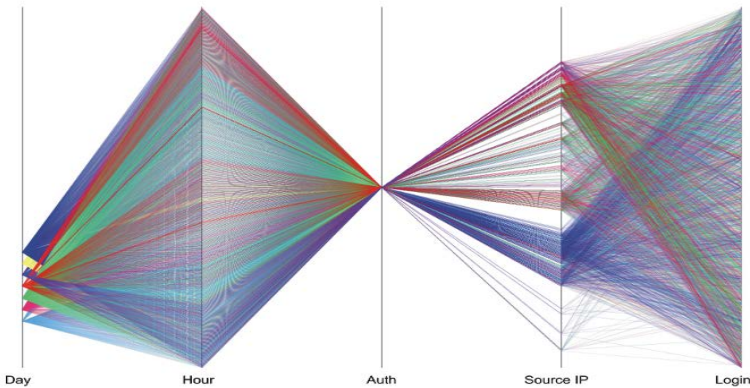


Fig. 5. SSH scanning over a six-day period on the most active days

Not only that, the coordination required is much more elaborate than previous botnets have shown in the past:

- A good variation of login ids, representing a coordinated dictionary attack, unfortunately the server was not prepared to record passwords in failed logins (that could help understand dictionary distribution over the bots);
- At most two attempts are launched each day from the same address.

In the Figure 5 that add some days when in Figure 4, demonstrates an effective grow in quantity of ips attempting to access the SSH server (in the bot of *Source IP* axis).

Filtering the ip axis to show only ip starting with 200 (200.*) and modifying the ip axis configuration (to spread ips along the axis) results on Figure 6 that reveal the uniqueness of the attack. The highlight in Figure 6 shows an absence of large fan-outs from each point on the *Hour* and *Source IP* axes, when compared to the detailed view in Figure 3.

With different colors for each day, we could easily see that many ips have one on the most two login attempts in the same day. It represent a very stealth scanning, that subvert standard configurations of defense tools like Denyhosts and Fail2ban. Distributed attacks and its derivations are difficult to detect and visualize with traditional defense tools [11].

PicViz-GUI was capable of showing a full view of the distributed SSH brute-force attack that was passing unnoticed by other means, revealing that present defense tools need improvements to detect and block this new approach. Shortly after this analysis was performed, news were seen [3,4] of large botnets actively exploiting a vulnerability in older versions of the phpMyAdmin tool (described in the *Common Vulnerabilities and Exposures* database under the number CVE-2009-1151).

By exploiting this vulnerability—in older phpMyAdmin versions—attackers were capable of executing malicious code on vulnerable hosts. The botnet herders exploit

³ Botnet Trend: phpMyAdmin & SSH Attacks – Malware city Website

⁴ SSH - new brute force tool – SANS ISC DIARIES

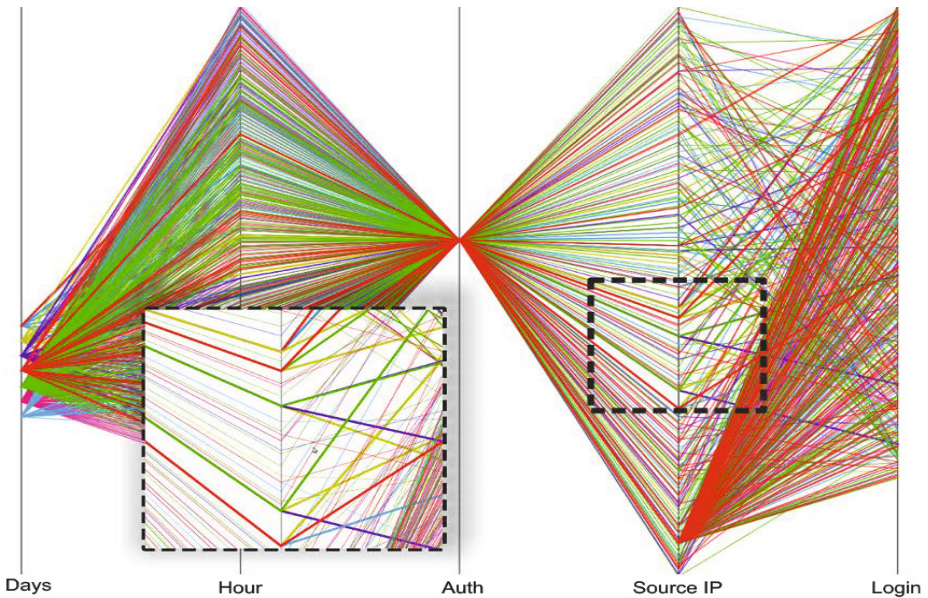


Fig. 6. Isolation of IPs from the 200 prefix range shows that individual hosts are trying remote accesses once a day (each day was colored differently)

this vulnerability and upload a nasty bot named “dd_ssh”; this bot then conducts a brute-force SSH scan attack on random IP addresses under the command of the herder.

The increasing number of source IP addresses noticed over the days in the plot is related to the number of hosts successfully infected by the CVE-2009-1151 vulnerability.

Picviz identified this new distributed SSH brute-force attack that went stealth to traditional defense tools. This tool allows quick identification of complex attack patterns that are not easily detectable and so allows for quicker mitigation and development of countermeasures.

Picviz-GUI present new possibilities for security visualization, but interactive visualization tools have to respect one line, maybe too much interaction could overwhelm user by a jumble of buttons, with no clear narrative path. The dynamic behavior of Picviz-GUI disposes a lot of benefits in addition to Picviz, the main is to avoid a lot of rounds of: tuning, generating and looking into images. These steps could be made with a few clicks.

Another problem of static images are the values in the chart, often the user needs to guess a value in the graph, because in some times putting labels for all points in the graph produces a misunderstanding picture. In addition, users can lose a lot of time simply by changing the colors of the data, because when choose the colors do not exactly know if they will overlap in the graph, and may get worse if the overlap of two colors create an undesired effect. Performed by the GUI, we simply select, create a layer and pick a color for that layer. The biggest advantage of static Picviz is the first insight, because we can look for a huge number of events, bigger than in the GUI. Maybe it is a primor step of the analysis in the whole framework.

The biggest advantage of static Picviz is the first insight, because we can look for a huge number of events, bigger than in the GUI. Maybe it is a obligatory first step of the analysis in the whole framework.

5 Related Work

Information Visualization is mature research field with a good range of studies that have been applied to many domains. Excelent surveys was written about this area [14] [15] [13], but recent work have been done to apply it to computer and network security. This new research field has increased over the past decade, the most of work done is described by a new security visualization survey [19].

The *ll-coordinates* technique is a well recognized method to understand multidimensional data, being already used in many research areas including computer security. Despite its large use base, most tools suffer from an ad-hoc motivation, focused on very specific data and not being flexible enough (for instance, being tied to pcap packet traces or to a specific log format).

The Krasser and Conti approach [5] uses *ll-coordinates* for realtime and forensic data analysis, combining *ll-coordinates* with time-sequence animation of scatter plots, but their limited scope prevents it to deal with a variety of attack classes. Flowtag [7] employs collaborative design to visual filtering and network flow tagging, a two axis *ll-coordinates* graph is used to represent connection ports and source ips, but this tool is focused in network traffic data. VisFlowConnect [18] draws *ll-coordinates* to help administrators understand the big picture in domain-wide behaviors based on network data flows.

Rumint is a parallel visualization tool designed to provide detailed insights into packet level network traffic, but not higher network levels [2]. In SHADOW [10], packet headers match pre-defined rules and then are dumped to a file for examination by a human operator. The authors describe different ways to plot this data.

By capitalizing on the PGDL structure and its dedicated data types, Picviz may be applied to any kind of information that can be parsed. In fact, given a suitable parser, Picviz could be used in other areas of Information Analisis.

6 Conclusion

This article described an interactive tool to analyze and document computer security incidents through *ll-coordinates*. Picviz provides simple mechanisms to interpret data from large data sets, thus allowing for an easier identification and understanding of complex events, such as those present in security log files.

We tested the tool on log files from the SSH service running on a host that offers remote access service to its users. Due to the amount of attacks directed to this service and constant new approaches employed by attackers to evade detection, these logs supplied the required data and at the same time buried the critical information that a security analyst should find. The data used were, however, good enough to explore the power of visualization tools when applied to security issues.

ll-coordinates provide a simple way to recognize information, whereas Picviz enhance that ability. Security log file information was processed to detect attack patterns against the SSH service that were not easily recognizable. The tool allowed the detection of both simple brute force scanning attacks and more sophisticated attacks launched through botnets. The latter could not be detected without a visualization tool that brought all the log information in a unique and consolidated view. The attackers behavior noticed during the analysis is of substantial importance for the improvement of tools aimed to detect and block such attacks.

References

1. Picviz homepage (2010)
2. Conti, G., Abdullah, K., Grizzard, J., Stasko, J., Copeland, J.A., Ahamad, M., Owen, H.L., Lee, C.: Countering security information overload through alert and packet visualization. *IEEE Computer Graphics and Applications* 26(2), 60–70 (2006)
3. da Silva Kauer, A.L., Meiguins, B.S., do Carmo, R.M.C., de Brito Garcia, M., Meiguins, A.S.G.: An information visualization tool with multiple coordinated views for network traffic analysis. In: 12th International Conference on Information Visualisation, IV 2008, pp. 151–156. IEEE (2008)
4. Inselberg, A., Dimsdale, B.: Parallel coordinates: a tool for visualizing multi-dimensional geometry. In: Proceedings of the 1st Conference on Visualization 1990, p. 378. IEEE Computer Society Press (1990)
5. Krasser, S., Conti, G., Grizzard, J., Gribshaw, J., Owen, H.: Real-time and forensic network data analysis using animated and coordinated visualization. In: Proceedings from the Sixth Annual IEEE SMC on Information Assurance Workshop, IAW 2005, pp. 42–49. IEEE (2005)
6. Kruse, W.G., Heiser, J.G.: Computer forensics: incident response essentials. Addison-Wesley (2008)
7. Lee, C.P., Copeland, J.A.: Flowtag: a collaborative attack-analysis, reporting, and sharing tool for security researchers. In: Proceedings of the 3rd International Workshop on Visualization for Computer Security, pp. 103–108. ACM (2006)
8. Notsu, H., Okada, Y., Akaishi, M., Nijima, K.: Time-tunnel: Visual analysis tool for time-series numerical data and its extension toward parallel coordinates. In: Proceedings of the International Conference on Computer Graphics, Imaging and Visualization, pp. 167–172. IEEE Computer Society (2005)
9. Ramsbrock, D., Berthier, R., Cukier, M.: Profiling attacker behavior following ssh compromises, pp. 119–124 (June 2007)
10. Solka, J.L., Marchette, D.J., Wallet, B.C.: Statistical visualization methods in intrusion detection. *Computing Science and Statistics* 32, 16–24 (2000)
11. Thames, J.L., Abler, R., Keeling, D.: A distributed active response architecture for preventing ssh dictionary attacks, pp. 84–89 (April 2008)
12. Tricaud, S., Saadé, P.: Applied parallel coordinates for logs and network traffic attack analysis. *Journal in Computer Virology* 6(1), 1–29 (2010)
13. Tufte, E.R., Goeler, N.H., Benson, R.: *Envisioning information*, vol. 21. Graphics Press Cheshire, CT (1990)
14. Tufte, E.R., Howard, G.: *The visual display of quantitative information*, vol. 7. Graphics press Cheshire, CT (1983)
15. Tufte, E.R., Weise Moeller, E.: *Visual explanations: images and quantities, evidence and narrative*. Graphics Press Cheshire, CT (1997)

16. Wegman, E.J.: Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 664–675 (1990)
17. Yang, J., Peng, W., Ward, M.O., Rundensteiner, E.A.: Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets (2003)
18. Yin, X., Yurcik, W., Treaster, M., Li, Y., Lakkaraju, K.: Visflowconnect: netflow visualizations of link relationships for security situational awareness. In: *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pp. 26–34. ACM (2004)
19. Zhang, Y., Xiao, Y., Chen, M., Zhang, J., Deng, H.: A survey of security visualization for computer network logs. In: *Security and Communication Networks* (2011)

Methodology for Detection and Restraint of P2P Applications in the Network

Rodrigo M.P. Silva and Ronaldo M. Salles

Military Institute of Engineering, Rio de Janeiro RJ, Brazil
rmpraxedes@yahoo.com.br, salles@ieee.org

Abstract. P2P networks are consuming more and more Internet resources, it is estimated that approximately 70% of all Internet carried traffic is composed by packets from these networks. Moreover, they still represent the main infection vector for various types of malware and can be used as command and control channel for P2P botnets, besides being famous for being notoriously used to distribute a range of pirated files (movies, music, games,...). In this paper we present some typical characteristics of P2P networks and propose a new architecture based on filters to detect hosts running P2P applications. We also provide a methodology on how to prevent the communication of those hosts in order to avoid undesirable impacts in the operation of the network as a whole.

1 Introduction

P2P networks replace the Client / Server architecture (centralized) used in traditional networks, for a decentralized topology consisting of hosts called peers that behave either as a server or a client, as illustrated in figure 1. The term Peer-to-Peer was invented by IBM in 1984 with the project Advanced Peer-to-Peer Networking Architecture, while P2P networks were later implemented through FidoNet [7] and UseNet [23]. However, it was only in the 90's with the popularization of the Internet, coupled with an increased availability of bandwidth for users and the advent of new technologies (e.g. mp3 (1995) and DivX (1999) that allowed high rates of compression of audio and video files), that these networks began to become popular and responsible for the distribution of content through applications such as Napster (1999), eDonkey (2000), Kademia (2002), among others [13,20].

Applications that use P2P networks, called P2P applications, have gained an increasing share of Internet traffic. Statistics from the Sourceforge site show that among the five most wanted applications, three are P2P applications, totaling almost one billion downloads. According to [12], it is estimated that approximately 70% of traffic is generated by this type of application.

Such growth has raised concerns on the part of ISPs, because although P2P applications are becoming the main source of online information, their rapid growth has degraded the quality of services provided by them. Moreover, the decentralized architecture of such networks together with the fact that there are no certification authorities to manage and control what is made on P2P networks,

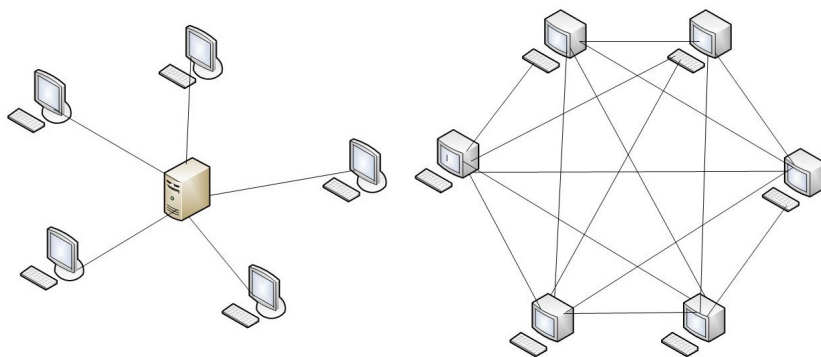


Fig. 1. Representation of a Client / Server topology (centralized) to the left and a P2P topology (decentralized) to the right

have turned them into an ideal environment for proliferation of pirated files (audio, video, games, etc). This unregulated behavior has alarmed the authorities that protect intellectual property, as the Recording Industry Association of America (RIAA) [4], and producers of content on governments and businesses.

Also, as a result of uncontrolled growth of these networks, it was observed an increased speed in which many malwares (worms, botnets, etc.) have spread. Wang et al. in [26] showed how botnets can use P2P networks as a vector for infestation, and use them as a Command and Control Channel. Lin et al. in [18] showed how the Distributed Hash Table (DHT) in a P2P network can be changed to spread some kinds of malware.

The contribution of this paper is threefold:

- we propose a novel architecture to detect hosts running P2P applications. Such architecture is based on a bank of filters that perform packet flow analysis using short time windows;
- we evaluate the performance of the proposed architecture and compare it with results obtained from previous approaches. Tests were carried out employing data sets containing traces from well-known repositories;
- we suggest possible network management actions regarding P2P traffic from the results obtained by the proposed architecture.

The rest of this paper is organized as follows. Existing works on P2P application detection are discussed in section 2. The adopted methodology and the system architecture is defined in section 3. In section 4, the data collection is discussed, the importance of the utilization of public repositories of traces to facilitate comparison among the most different works. In section 5, we show results obtained and in section 6 a comparison with previous approaches. In Section 7, we highlight some network management actions that may be used to support security policies. Finally, the paper is concluded in section 8 together with comments about future work.

2 Related Works

Initially, the most applied scheme to detect P2P applications was to check port numbers in traffic packets [16,15]. Unfortunately modern applications in order to circumvent this technique have adopted a great flexibility in the choice of ports to be used, and can even use well known ports reserved for other typical applications to avoid being detected by firewalls.

Signature-based techniques have also been widely used, they work by looking for patterns inserted in the payload to distinguish P2P packets from others. Such techniques, as used in [9,21] offer low percentage of false positives, however it is possible to point out several deficiencies in relation to existing networks and applications, as follows:

- signature-based techniques do not have a good performance in networks with high bandwidth due to the cost of processing and analysis packet contents;
- not all packet contents are available for analysis, often only the header is available, as shown in [5]. Data traffic over networks tends to be treated as confidential information and leaks can result in serious legal implications;
- current P2P applications use encryption to avoid detection, making payload analysis inefficient.

Moreover, it can be noted that both techniques under consideration, based on port numbers and payload analysis, do not allow the detection of new P2P protocols, limiting detection to applications that use known protocols.

In [16], techniques were proposed to use the characteristics of the data flow among nodes in the analysis. Such techniques look for inherent patterns related to the behavior of P2P protocols, which made it more difficult for the application to change behavior and circumvent all its features trying to evade detection. Unfortunately with the advent of hosts with more memory and processing capacity, the number of services running on the same host has largely increased, which may hinder the analysis of these patterns in the network.

In [24], Spognard et al. focused their work in specific patterns of P2P protocol that can be revealed by an IP packet level analysis. Such patterns are coded into rules that are fed to an IDS. The main limitation of their work is that some simple alterations in the P2P application can easily avoid the proposed detection technique.

Liu et al. in [19], developed a new method of P2P traffic identification based on Support Vector Machine (SVM). The method was implemented by analyzing packet lengths, remote hosts discreteness, connection success rate and the ratio of IP and Port numbers at the host level. However, the method presented false negative rates of up to 12% in some cases according to the authors, which may be considered high for such a sophisticated approach.

The work in [14] presented a P2P detection technique based on Traffic Dispersion Graphs (TDG). The technique searches for inherent patterns in P2P protocols in the generated graphs and focuses on features of graph metrics instead of individual flows. Despite having high accuracy it demands high processing

time to return a solution, which makes such approach not appropriate for online detection.

In [17], Kim et al. presented methods for detecting P2P flows by constructing a graph where each flow is a vertex. Edges are constructed by applying various rules that consider the ports used by previously detected P2P flows. In the graph, they found that around 90% of the P2P flows are within a large connected component, the remaining part is composed of many smaller connected components. Like other techniques based on graphs, this demands high processing times.

3 The Proposed P2P Detection Architecture

The architecture proposed in this work is illustrated in figure 2. It includes a host agent, directly linked to the network router, which is responsible for collecting all packets originating from and destined to the network, converting them into flows and store them in database tables. All this process and data is limited to a 7 minutes' interval. Tables are filtered out and the resulting "P2P" hosts (hosts running P2P applications) are registered in firewalls in order to block/limit such communication, according to the policies applied by the network administration.

Our premise is to use short time windows for analysis, because of the dynamics of networks, where nodes are constantly being swapped in and out of the network. This requires the analysis to be done as soon as possible to get information in time to be useful, e.g. take some immediate action.

One of the most important steps in our method is filtering: the process of separating and classifying the collected data into P2P or non-P2P traffic. Our proposal is to combine a set of filters to enhance the capacity detection of the approach. In the proposed architecture, as seen in figure 2, all data flows pass through a DNS filter and then to a bank of filters. Marked flows are used by the decision maker to set up firewall rules and/or to apply the security policies established by administration authorities.

We analyzed the work of several previous authors and select the most promising filters to compose our architecture. Such filters are discussed in greater detail in the next subsection.

3.1 Analyzed Filters

All filters used were evaluated in previous works by several authors.

- **DNS filter:** Zhang et al. in [27] used the DNS requests to reduce the flow volume to be analyzed, this is due to the fact that P2P communication is carried out directly through the IP addresses of hosts members in peer lists of networks. The percentage of peers with domains registered in DNS is less than 0.5%, according to the authors. By means of analysis of traces of P2P and non-P2P applications it was proven that it is possible to reach a reduction of more than 25% of the total flow of the network to be analyzed (figure 3).

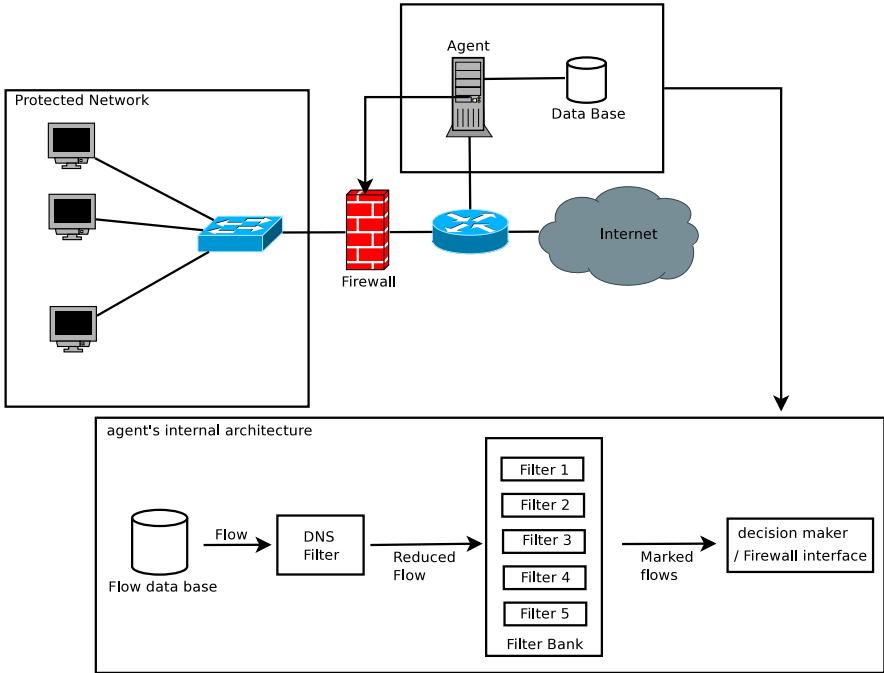


Fig. 2. Proposed architecture

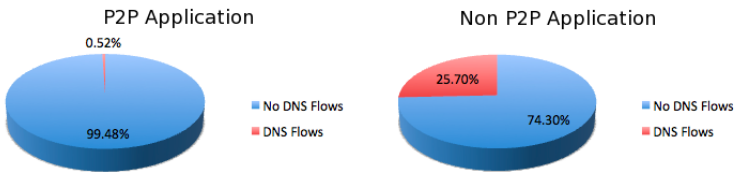


Fig. 3. Using DNS for P2P and non-P2P applications

- **Filter 1. Sending rate of TCP SYN / UDP:** Chen et al. in [8], showed that when a new node tries to join the established P2P network, it sends TCP SYN packets to different IP addresses so as to find the best entry to the network. During this period of time, TCP SYN packets provide a notable behavior. If a specific IP address sends lots of TCP SYN packets to many different IP addresses, it can be identified as P2P communications. This behavior is also valid for UDP connections, except to the fact that there is no three way handshake.
- **Filter 2. Rate of successful connections:** [27][25] showed that because the majority of peers are not servers, they can be turned off at certain times of day, or even can be with other IP addresses, behind firewalls or NAT servers. This reduces the chance of other peers to be able to connect,

thereby reducing the rate of successful connections when compared to other services based on the client / server topology.

- **Filter 3. Rate IP_{ext} / $Port_{ext}$:** in [8][25] it was shown that when a P2P client joins the distributed network, it establishes several connections with other peers (assume as n) on different ports (assume as m). The authors found that $n \simeq m$. Thus when a fix pair composed by IP and Port builds connection with different peers and the number of different IPs is approximately equal to the number of different ports (the rate is approximately 1), the P2P communication channel can be confirmed.
- **Filter 4. Port number:** Ulliac et al. [25] used port number information as a feature to detect hosts that run P2P applications. Initially each type of application was associated to fixed port numbers and such mapping could be used to detect application type. However, in order to evade simple detection mechanisms, P2P applications tend to pick random ports, as shown by Kariaginnis et al. in [16]. A host to receive messages from other peers must select a port, as there are 65,536 ports the likelihood of a given port be chosen is approximately 0.0015%, which leads to a chance of approximately 98.5% that a P2P application choose a port above 1024.
- **Filter 5 (new). Analysis of source ports:** a new feature is proposed in this paper, P2P applications use a single source port to make connections with other peers. This fact is illustrated in figure 4, which shows two hosts with and without P2P applications, where each vertex indicates an IP:Port pair and each edge indicates the communication made. A host running P2P application opens a large number of calls to several other nodes using the same source port, a characteristic that can be explored to distinguish hosts with P2P applications from others. Moreover, this feature allows to create a specific firewall rule that considers only the used P2P ports and thus avoiding inconveniences to the user's host (other application from the same host will not be blocked by such firewall rules).

Figure 5 shows a flow related to an IP host running P2P applications. It is possible to see the existence of several UDP sessions originating from the same port to several different IP addresses, such sessions are peer attempts to establish communication in the P2P network.

4 Data Collection

The results shown in this paper were reached using data from existing repositories of traces on the Internet. As shown in [5], one of the main difficulties encountered by researchers in the area is to obtain consistent traces, which are able to show the heterogeneous reality found on the Internet. Several research results are based only in academic network traces, which are generated from machines that are presumably located in a controlled environment, differently from those found in network companies managed by ISPs. Another widespread alternative for the validation test is to use synthetic data, however such approach is also limited to represent all characteristics of P2P applications, compromising the validation of the proposed methodology.

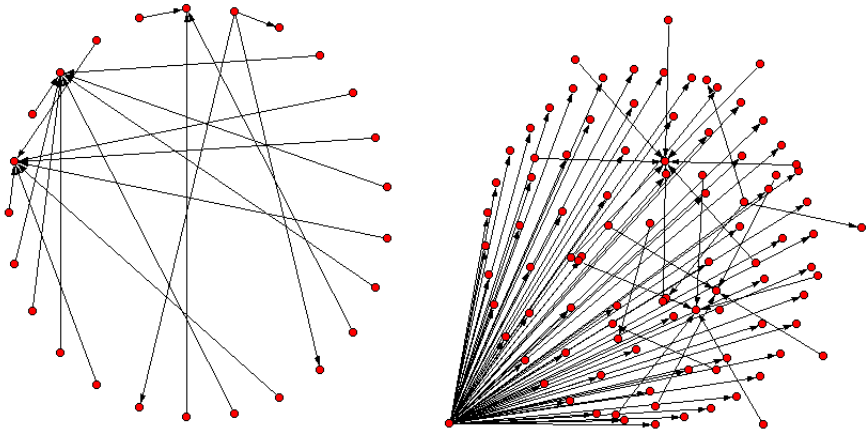


Fig. 4. Linking $IP : Port_{src}$ and $IP : Port_{dest}$ between hosts without P2P application (left) and between hosts with P2P applications (right)

1268823396.296315	15.230969	udp	85.189.119.122	57362	<->	182.169.43.216	6881
1268823396.321759	728.668640	udp	85.189.119.122	57362	<->	92.253.34.96	49555
1268823396.347074	790.515320	udp	85.189.119.122	57362	<->	252.220.182.94	26332
1268823396.372552	750.832458	udp	85.189.119.122	57362	<->	95.105.159.114	60243
1268823396.397927	1.001678	udp	85.189.119.122	57362	->	78.43.171.225	39654
1268823396.423335	80.075356	udp	85.189.119.122	57362	<->	190.31.10.214	13703
1268823396.448712	11.731586	udp	85.189.119.122	57362	<->	215.105.178.194	31290
1268823396.474117	717.640503	udp	85.189.119.122	57362	<->	76.185.127.234	20296
1268823396.499511	781.028931	udp	85.189.119.122	57362	<->	87.93.35.94	33031
1268823396.524751	80.069145	udp	85.189.119.122	57362	<->	84.229.79.228	21506
1268823396.550367	30.928553	udp	85.189.119.122	57362	<->	119.39.119.113	54963
1268823396.575676	1100.265381	udp	85.189.119.122	57362	<->	118.57.166.86	18879

Fig. 5. Trace showing the connections made from some source ports of a peer (85.189.119.122) to other peers. Columns represent respectively: timestamp , duration, protocol, source address, source port, direction, destination address, destination port.

Another problem related to the use of traces is that it is often difficult to obtain accurate information about its content. The trace should be completely marked and fully known, otherwise any performance evaluation result (e.g. false positive and false negative rates) could be compromised. Most real traces carry unknown traffic patterns from diverse applications and services. In addition to that, when comparing a novel technique with previous approaches it is important

to employ traces from stable repositories and design a standard script of testing to serve as a benchmark and validate the results obtained.

5 Results

We used four different data sets to perform tests on the proposed architecture,

- D_1 : contains three traces of a host running a bittorrent application and was obtained from the repository CRAWDDAD [2];
- D_2 : two traces of a network with no P2P application, such trace was taken from DARPA [1] and OpenPacket [3] repositories;
- D_3 : was also obtained from the CRAWDDAD repository and belongs to a network that has online game sessions;
- D_4 : other traces from the OpenPacket repository containing hosts with and without P2P applications.

Table 1. Data sets used in the tests

Set	Traces	Source
D_1	3	CRAWDDAD
D_2	2	DARPA / OpenPacket
D_3	2	CRAWDDAD
D_4	2	OpenPacket

All traces were converted into flows using Argus Auditing Network Activity (<http://www.qosient.com/argus/>) and stored in MySQL database tables that were rotated every 7 minutes. These tables were analyzed separately by the proposed architecture and hosts marked as potential members of P2P networks were added in an iptables firewall. The results are summarized in Table 2.

5.1 Set D_1 : Traces with P2P Applications

This set of traces contains host with P2P applications. It was observed a DNS percental use below 0.5%, which confirms what was stated in [27] that peers do not use DNS to get the addresses of the other peers, such addresses are recorded in peer lists in that host. Thus confirming that the exclusion of any flow that is related to DNS queries does not jeopardize the detection of P2P applications.

Regarding the rate of TCP / UDP connections made by network hosts we observed that P2P hosts presented rates above 60 connections / minute when using UDP protocol, which is explained by the fact that P2P applications, as shown in [22,11], use UDP to exchange control messages (make new connections, update parameters related to peers or even maintaining the network in operation). On the other hand, the rate of TCP connections is lower around 16 connections / minute , as this protocol is mostly used for file transfer only.

The rate of unsuccessful connections is also quite high in this type of application reaching around 35% in some cases. This is due to the fact that many peers are not online, are using other IP addresses or are even behind firewalls, which prevents that connections are made.

The ratio between the number of external ports and external IP addresses connected to each of the P2P hosts was close to 1, ranging up to 1.23. Regarding the use of the same source port, it could be seen that approximately 78% of the flows generated are originated from the same port.

The average value of port numbers is also an excellent detection criterium, since current P2P applications tend to use random ports for communication, as shown by [16]. In our samples the average values were all above 45,000.

5.2 Set D_2 : Traces without P2P Applications

The dataset D_2 consists of traces without P2P applications. We observed a DNS access rate greater than the previous set, with values above 14%.

The average rate of connections was less than 10 connections / minute. Regarding the rate of unsuccessful connections, the value obtained was less than 10%, however two hosts presented higher rates (above 40%). This behavior was motivated by the high number of requests on ports 137 and 139, both used by the NetBIOS Name service.

The average ratio of the number of connected external ports and the number of IP addresses is greater than 11 for the hosts in the set. The percentage of source ports used for connections was similar, the largest percentage obtained was 8% for a given port.

The average value of the destination ports used was 3,231.

5.3 Set D_3 : Online Games

This set contains traces related to exchanges of messages between hosts during the execution of online games. It differs from the other sets by several distinctive features.

Regarding the use of DNS, it is clear that at no time queries are sent to a DNS, due to the fact that the hosts of the game session know the IP address of other hosts. Average rate of connections is less than 2 connections / minute, for the reason that the number of participants per session is limited and defined, which avoids hosts session members trying to contact other hosts. The rate of unsuccessful connections is 7% failure, probably motivated by sudden disconnections.

The ratio between the number of external ports and external IP addresses presented a high variation making this feature not interesting to be used in traffic identification. Regarding source ports, the percentage of use was well balanced among all ports the traces, no prominence of any port was observed.

The set also has an average value of used ports very high, similar to what happens in P2P applications. This fact is justified by the attempt of many online

games to avoid firewall detection mechanisms through the use of random ports for communication.

5.4 Set D_4 : Traces with and without P2P Applications

In order to better test the performance of the proposed methodology we used traces containing hosts with and without P2P applications.

About the use of DNS, the average value obtained for normal hosts was approximately 16%, while for hosts with P2P applications was around 1.3%. The average rate of connection for normal hosts was around 2.5 connections / minute, whereas for P2P hosts that rate reached values above 20 connections / minute.

Regarding the percentage of connection failures, hosts with P2P applications behaved with an average value above 30%, while other hosts showed a failure rate lesser than 5%. The ratio of the number of external ports and external IP address was above 2 for hosts without P2P application and was approximately 2 for hosts with P2P application.

Analyzing the percentage of connections made by source ports, it was observed that more than 25% of all network connections were made by a single port of the host with P2P application. In addition, 8% of the connections were made from a port of a second host without P2P application. All other hosts had a relatively balanced proportion with no special cases.

The average value of ports used for hosts without P2P application was 8,212, while for hosts with P2P applications was 15,581.

Table 2. Mean values found for each data set

Set	Using DNS	connections rate	Connections unsuccessful	Rate $Port_{ext}/IP_{ext}$	Average value of ports	Connections $Port_{src}$
D_1	0.43%	65 conn / min	35%	1,1	48,215	78% (main port)
D_2	14.2%	6 conn /min	9%	11.6	3,231	8% (no emphasis)
D_3	0%	1.8 conn / min	3.5%	1.8	26,323	7% (no emphasis)
D_4	15.4%	2.5 conn /min	4.5%	2.4	8,212	8% (no emphasis)
D_4 (host P2P)	1.3%	19 conn / min	33.5%	1.8	15,581	25% (main port)

6 Comparison with Previous Works

In this section, we aim to draw a comparison on the performance achieved by the proposed solution and the various solutions proposed in previous works. The criterion for comparison is based on the following attributes: time required for detection, false positive rate, scalability and analyzed features.

In [16], Karagiannis et al. were the first to use identification of P2P applications based on features of the transport layer (connection rate, protocols). The time required for detection depends on the data flow. According to the authors, the smaller the number of flows analyzed the higher the rate of false positives. In that paper the false positive rate obtained was about 5%. Such a system given its properties and construction is not scalable.

In [25], Ulliac et al. defined a detection methodology in two stages. In the first stage filters were used to discuss features of applications (average value of the ports, standard deviation of the value of ports) and connections (average number of connections per host, the average rate of connections). In the following step, traffic features are analyzed (packet rate). Regarding the rate of false positives the authors reached a value of 15% in real networks. The algorithm is not scalable since its accuracy is reduced as the network traffic intensifies.

Bo et al. in [6] developed a distributed architecture where several hosts located in different networks are responsible for detecting P2P applications on their networks and share information with others to assist them in the identification. The main feature used for detection is the connection rate performed by each host. This method can be used to support other architectures to improve their scalability and accuracy. Because the detection is based on number of connections in a short period of time, it can be considered scalable.

Liu et al., in [19] developed a method based on Support Vector Machine (SVM) for the decision on which traffic is originating from hosts with P2P applications. The authors analyzed features about connections among hosts (number of connections per host, rate of failed requests). The percentage of false positives in their proposal reached 7.3%. Moreover, due to the need for processing the SVM, this system may require a cluster to be scalable.

In [10], Chunzhi et al. developed a method based on a decision tree that examines features of P2P applications (like ration between TCP and UDP protocols and packet ratio change) and traffic. The false positive rate was around 3%. The

Table 3. Comparison among methods for detection of P2P applications

Method	Detection time	Analyzed features	Scalability	Percentage of false positives
[16]	depends on the flow on the network	connection	no	5%
[25]	short	applications, connections	no	15%
[6]	short	connections	yes	-
[19]	short	connections	yes (clustering)	7.3%
[10]	short	signature, traffic	yes (clustering)	3%
Current proposal	short	connections, applications, traffic	yes (clustering)	3%

processing time is dependent on the packet flow in the network. The method can be considered scalable provided the adequate infrastructure is employed.

In this paper, we studied a methodology based on different sets of features trying to encompass all important inherent behavior of a P2P application. The method uses a 7 minute-window of duration to perform flow detection. The rate of false positives obtained from the traces was around 3%.

7 Proposed Methodology to Restrain P2P Traffic

Given the results presented in previous sections, we propose a simple methodology based on levels of alert, as shown in table 4.

Level 0 indicates that no suspicious behavior has been identified therefore nothing is done. At levels 1 and 2 some suspicious behavior has been detected and a message is sent to network manager containing the IP address and a suggestion according to the respective level: "detailed analysis of the behavior of the host" or "insertion of the host in the firewall rules". At level 4 the host is immediately inserted in the firewall rules and a message is sent to inform the network manager.

Table 4. Alert levels and corresponding actions

Level	Description	Procedure
0	no feature is found	nothing is done
1	one feature is found	send message suggesting a detailed analysis of the host behavior
2	two or three features are found	send message suggesting the insertion of the host in the firewall rules
3	four or more features are found	automatic insertion of the host in the firewall rules and send alert message

8 Conclusion and Future Works

In this paper we discuss how P2P applications are constantly growing, generating a series of problems in the networks they inhabit. In view of this situation we proposed a methodology based on mining of packet flows in networks to detect the existence of such applications. Some intrinsic characteristics of these applications were analyzed and a set of filters were implemented in the proposed architecture to construct a consistent detection scheme.

Some tests were performed using selected traces. In response to those tests, hosts with P2P applications had their outstanding features revealed. However, it was also shown that some hosts that run competing services had such features distorted, making detection more difficult.

In order to have a method as accurate as possible, each feature was analyzed in parallel in each host using different short time windows. The proposed architecture provided a good degree of accuracy and detection.

In order to continue the work, the indices of false positives and negatives will be analyzed in greater detail, as well as its variation when short time windows are changed in the network.

Furthermore, we intend to look into the differences between networks of P2P applications and P2P botnets aiming to adapt this technique to detect this type of malware.

References

1. Intrusion detection evaluation (1999), <http://www.ll.mit.edu>
2. A community resource for archiving wireless data at dartmouth, (2012), <http://crawdad.cs.dartmouth.edu>
3. Open packet (2012), <https://www.openpacket.org>
4. Recording industry association of america (2012), <http://www.riaa.com>, <http://www.riaa.com/physicalpiracy.php>
5. Aviv, A.J., Haeberlen, A.: Challenges in experimenting with botnet detection systems. In: Proceedings of the 4th USENIX Workshop on Cyber Security Experimentation and Test (CSET 2011) (2011)
6. Bo, X., Ming, C., Lan, F.: Distributed p2p traffic identification method. In: Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2009, pp. 4229–4232. IEEE Press, Piscataway (2009), <http://dl.acm.org/citation.cfm?id=1738467.1738494>
7. Bush, R.: Fidonet: technology, tools, and history. *Commun. ACM* 36, 31–35 (1993), <http://doi.acm.org/10.1145/163381.163383>
8. Chen, F., Wang, M., Fu, Y., Zeng, J.: New detection of peer-to-peer controlled bots on the host. In: 5th International Conference on Wireless Communications, Networking and Mobile Computing, WiCom 2009, pp. 1–4 (September 2009)
9. Choi, T., Kim, C., Yoon, S., Park, J., Lee, B., Kim, H., Chung, H., Jeong, T.: Content-aware internet application traffic measurement and analysis. In: IEEE/IFIP Network Operations and Management Symposium, NOMS 2004, vol. 1, pp. 511–524 (April 2004)
10. Chunzhi, W., Wei, J., Hong, C., Luo, W., Fang, H.: Research on a method of p2p traffic identification based on multi-dimension characteristics. In: 2010 5th International Conference on Computer Science and Education (ICCSE), pp. 1010–1013 (August 2010)
11. Erman, D., Ilie, D., Popescu, A.: Bittorrent session characteristics and models. In: Proceedings of HETNETS 2005, p. 2007 (2005)
12. Erman, J., Mahanti, A., Arlitt, M., Williamson, C.: Identifying and discriminating between web and peer-to-peer traffic in the network core. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, pp. 883–892. ACM, New York (2007), <http://doi.acm.org/10.1145/1242572.1242692>
13. Hong, S.H.: Measuring the effect of napster on recorded music sales: Difference-in-differences estimates under compositional changes. *Journal of Applied Econometrics*, 1–28 (2011), <http://dx.doi.org/10.1002/jae.1269>
14. Iliofotou, M., Kim, H.C., Faloutsos, M., Mitzenmacher, M., Pappu, P., Varghese, G.: Graption: A graph-based p2p traffic classification framework for the internet backbone. *Computer Networks* 55(8), 1909–1920 (2011), <http://linkinghub.elsevier.com/retrieve/pii/S1389128611000430>

15. Karagiannis, T., Broido, A., Brownlee, N., Claffy, K., Faloutsos, M.: Is p2p dying or just hiding (p2p traffic measurement). In: IEEE Global Telecommunications Conference, GLOBECOM 2004, November–December 3, vol. 3, pp. 1532–1538 (2004)
16. Karagiannis, T., Broido, A., Faloutsos, M., Claffy, K.: Transport layer identification of p2p traffic. In: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement, IMC 2004, pp. 121–134. ACM, New York (2004), <http://doi.acm.org/10.1145/1028788.1028804>
17. Kim, J., Shah, K., Bohacek, S.: Detecting p2p traffic from the p2p flow graph. In: IWCMC, pp. 1795–1800. IEEE (2011), <http://dblp.uni-trier.de/db/conf/iwcmc/iwcmc2011.html#KimSB11>
18. Lin, H., Ma, R., Guo, L., Zhang, P., Chen, X.: Conducting routing table poisoning attack in dht networks. In: International Conference on Communications, Circuits and Systems (ICCCAS), pp. 254–258 (July 2010)
19. Liu, F., Li, Z., Nie, Q.: A new method of p2p traffic identification based on support vector machine at the host level. In: International Conference on Information Technology and Computer Science, ITCS 2009, vol. 2, pp. 579–582 (July 2009)
20. Locher, T., Mysicka, D., Schmid, S., Wattenhofer, R.: A peer activity study in edonkey & kad (1995)
21. Moore, A.W., Papagiannaki, K.: Toward the Accurate Identification of Network Applications. In: Dovrolis, C. (ed.) PAM 2005. LNCS, vol. 3431, pp. 41–54. Springer, Heidelberg (2005)
22. Ripeanu, M.: Peer-to-peer architecture case study: Gnutella network. In: Proceedings of First International Conference on Peer-to-Peer Computing, pp. 99–100 (August 2001)
23. Sit, E., Morris, R., Kaashoek, M.F.: Usenetdht: a low-overhead design for usenet. In: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2008, pp. 133–146. USENIX Association, Berkeley (2008), <http://dl.acm.org/citation.cfm?id=1387589.1387599>
24. Spognardi, A., Lucarelli, A., Di Pietro, R.: A methodology for p2p file-sharing traffic detection. In: Second International Workshop on Hot Topics in Peer-to-Peer Systems, HOT-P2P 2005, pp. 52–61 (July 2005)
25. Ulliac, A., Ghita, B.V.: Non-intrusive identification of peer-to-peer traffic. In: Proceedings of the 2010 Third International Conference on Communication Theory, Reliability, and Quality of Service, CTRQ 2010, pp. 116–121. IEEE Computer Society, Washington, DC (2010), <http://dx.doi.org/10.1109/CTRQ.2010.27>
26. Wang, P., Wu, L., Aslam, B., Zou, C.: A systematic study on Peer-to-Peer botnets. In: Proceedings of 18th International Conference on Computer Communications and Networks, ICCCN 2009, pp. 1–8 (August 2009)
27. Zhang, J., Perdisci, R., Lee, W., Sarfraz, U., Luo, X.: Detecting stealthy p2p botnets using statistical traffic fingerprints. In: International Conference on Dependable Systems and Networks, pp. 121–132 (2011)

Text Categorization Based on Fuzzy Soft Set Theory

Bana Handaga and Mustafa Mat Deris

Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia

handaga.bana@gmail.com, mmustafa@uthm.edu.my

Abstract. In this paper, we proposed a new method for Text Categorization based on fuzzy soft set theory so called fuzzy soft set classifier (FSSC). We use fuzzy soft set representation that derived from the bag-of-words representation and define each term as a distinct word in the set of words of the document collection. The FSSC categorize each document by using fuzzy c-means formula for classification, and use fuzzy soft set similarity to measure distance between two documents. We perform the experiments with the standard Reuters-21578 dataset, and using three kind of weighing such as boolean, term frequency, and term frequency-invert document frequency to compare the performance of FSSC with others four classifier such as kNN, Bayesian, Rocchio, and SVM. We are using precision, recall, F-measure, return-size, and the running time as a performance evaluation. Result shown that there is no absolute winner. The FSSC has precision, recall, and F-measure lower than SVM, and kNN but FSSC can work faster than both. When compared with the Bayesian and Rocchio, the FSSC works more slowly but has a higher precision and F-measure.

Keywords: Fuzzy soft set theory, bag-of-words, Text Classification.

1 Introduction

In the last 20 years content-based document management tasks, collectively known as information retrieval (IR), have gained a prominent status in the information systems field, due to the increased availability of documents in digital form and the ensuing need to access them in flexible ways. Text categorization (TC), the activity of labeling natural language texts with thematic categories from a predefined set, is one such task. TC dates back to the early '60s, but only in the early '90s did it become a major subfield of the information systems discipline, thanks to increased applicative interest and to the availability of more powerful hardware. TC is now being applied in many contexts, ranging from document indexing based on a controlled vocabulary, to document filtering, automated metadata generation, word sense disambiguation, population of hierarchical catalogues of Web resources, and in general any application requiring document organization or selective and adaptive document dispatching [1].

Many learning algorithms such as k-nearest neighbor, Support Vector Machines (SVM) [2-4], neural networks [5], fuzzy set [6], intuitionistic fuzzy sets [7], rough set

[8], linear least squares fit, and Naive Bayes [9] have been applied to TC. A comparison of these techniques is presented in [10].

Classification based on soft-set theory was made by Mushrif et. al. [11], using the principle of scoring that has been developed by Roy and Maji [12, 13] in fuzzy soft set theoretic approach to decision making problems. Unfortunately, this algorithm can not be applied to classify the text document.

In this paper, we proposed a new method for TC based on fuzzy soft set theory, using the idea of fuzzy c-means clustering [14] but it is used for classification, using this method every document has a degree of membership in all classes of documents. A document will be classified into a particular class if the document has a certain degree of membership in that class. Formula to calculate the degree of membership of a particular document on the document class, similar to the formula for calculating the degree of membership in the fuzzy c-means, but the distance between documents is measured using the principle of fuzzy similarity[15].

The paper is organized as follows: Section 2 discusses the related work and section 3 discusses the document representation. Section 4 presents the notions of fuzzy soft set theory and relevant definitions used in the proposed work and Section 5 gives an overview of the fuzzy soft set classifier. In Section 6, we describe the standard Reuters-21578 dataset we have used in the experiments, our experimental methodology, evaluation metrics, and the results we have obtained. We conclude in Section 7.

2 Related Work

Most of the studies aimed at solving the TC problem implement the bow structure. Using a machine learning algorithm that considers the terms in the training and test data as the basic features is the fundamental and conventional architecture for the text classification problem [10, 16]. In this approach, documents are represented by the widely-used vectorspace model introduced by [17]. Each document is represented as a vector d . Each dimension in the vector d stands for a distinct term (word) in the term space of the document collection based on the bow approach. Representing the terms in this way causes the word ordering information within the sentences to be lost.

String kernels with n-gram sequences were proposed to compensate for the ordering information and yielded promising results [18]. But this method has to deal with performance problems in large datasets it suffers big space and time complexities and thus uses approximation algorithms instead of representing the full structure.

A different approach is making use of a language model (representing a document by the generation of new sentences from the document itself based on finite automata and probabilistic models) for text categorization. Language models are sophisticated approaches used in information retrieval and they are accepted as too complicated models for text classification [16]. These models are more appropriate for problems like query generation from texts, speech recognition, etc.

Main machine learning approaches used in the TC domain may be classified as supervised (e.g. support vector machine) vs. semisupervised (e.g. using naive Bayes with expectation maximization) methods, parametric (e.g. support vector machine,

naive Bayes) vs. non-parametric (e.g. k-nearest neighbor) methods, linear (e.g. support vector machine with linear kernel) vs. non-linear (e.g. support vector machine with radial basis kernel) classifiers, vector space (e.g. artificial neural network, Rocchio) vs. probabilistic (e.g. naive Bayes) classification, and decision tree modeling (e.g. rule-based decision trees). Clustering (e.g. k-means, which is unsupervised and semiparametric) may also be employed in the case of the existence of a dataset without labeled training data. Several studies have compared the performances of these approaches and in general support vector machine (SVM) with linear kernel was shown to yield the leading results [2, 4, 10, 19]. For the fundamental challenges in the text classification domain (high dimensionality, sparse instances, separability of classes), SVM provides efficient solutions by being more immune to the overfitting problem, using an additive algorithm with an inductive bias that suits problems with dense concepts and sparse instances, and employing a basic linear separation model that fits the discrimination of most of the classes [4].

3 Document Representation

Documents should first be transformed into a representation suitable for the classification algorithms to be applied. In our study, documents are represented by the widely used vector-space model, introduced by [17]. In this model, each document is represented as a vector d . Each dimension in the vector d stands for a distinct term in the term space of the document collection. We use the bag-of-words representation and define each term as a distinct word in the set of words of the document collection.

To obtain the document vectors, each document is parsed, non-alphabetic characters and mark-up tags are discarded, case-folding is performed (i.e. all characters are converted to the same case-to lower case), and stopwords (i.e. words such as “an”, “the”, “they” that are very frequent and do not have discriminating power) are eliminated. We use the list of 571 stopwords used in the Smart system [17, 20]. In order to define words that are in the same context with the same term and consequently to reduce dimensionality, we stem the words by using Porters Stemming Algorithm [21], which is a commonly used algorithm for word stemming in English. We represent each document vector as $d = (w_1, w_2, \dots, w_n)$ where w_i is the weight of i th term of document d .

There are various term weighting approaches studied in the literature [22]. Boolean weighting and tf-idf (term frequencyinverted document frequency) weighting are two of the most commonly used ones.

In boolean weighting, the weight of a term is considered to be 1 if the term appears in the document and it is considered to be 0 if the term does not appear in the document:

$$w_i = \begin{cases} 1, & \text{if } tf_i > 0 \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

where tf_i is the raw frequency of term i in document d , tf-idf weighting scheme is defined as follows:

$$w_i = tf_i \cdot \log \left(\frac{n}{n_i} \right) \quad (2)$$

where tf_i is the same as above, n is the total number of documents in the document corpus and n_i is the number of documents in the corpus where term i appears. tf-idf weighting approach weights the frequency of a term in a document with a factor that discounts its importance if it appears in most of the documents, as in this case the term is assumed to have little discriminating power. Also, to account for documents of different lengths we normalize each document vector so that it is of unit length.

In his extensive study of feature selection metrics for SVM-based text classification, Forman used only boolean weighting [19]. However, the comparative study of different term weighting approaches in automatic text retrieval performed by Salton and Buckley reveals that the commonly used tf-idf weighting outperforms boolean weighting [22]. On the other hand, boolean weighting has the advantages of being very simple and requiring less memory. This is especially important in the high dimensional text domain. In the case of scarce memory resources, less memory requirement also leads to less classification time. Thus, in our study, we used both the boolean weighting and the tf-idf weighting schemes.

4 Soft Set Theory

In this section, we recall the basic notions of soft sets and fuzzy soft sets. Let U be an initial universe of objects and E_U (simply denoted by E) the set of parameters in relation to objects in U . Parameters are often attributes, characteristics, or properties of objects. Let $P(U)$ denote the power set of U and $A \subseteq E$. Following [23, 24], the concept of soft set is defined as follows.

Definition 1. ([24]) Let U be initial universal set and let E be a set of parameters. Let $P(U)$ denote the power set of U . A pair (F, E) is called a soft set over U , if only if F is a mapping given by $F:A \rightarrow P(U)$.

By definition, a soft set (F, E) over the universe U can be regarded as a parameterized family of subsets of the universe U , which gives an approximate (soft) description of the objects in U .

4.1 Fuzzy Soft Set

Definition 2. ([25]) Let U be an initial universal set and let E be set of parameters. Let $\tilde{P}(U)$ denote the power set of all fuzzy subsets of U . Let $A \subset E$. A pair (\tilde{F}, E) is called a fuzzy soft set over U , where \tilde{F} is a mapping given by $\tilde{F} : A \rightarrow \tilde{P}(U)$.

In the above definition, fuzzy subsets in the universe U are used as substitutes for the crisp subsets of U . Hence it is easy to see that every (classical) soft set may be considered as a fuzzy soft set. Generally speaking $\tilde{F}(\mathcal{E})$ is a fuzzy subset in U and it is called the fuzzy approximate value set of the parameter \mathcal{E} .

4.2 Similarity between Two Fuzzy Soft Sets

Measuring similarity or distance between two entities is a key step for several data mining and knowledge discovering task, such as classification and clustering. Similarity measures quantify the extent to which different patterns, signals, images or sets are alike. Several researchers have studied the problem of similarity measurement between fuzzy sets, fuzzy numbers and vague sets. Recently, Majumdar and Samanta [26] have studied the similarity measure of fuzzy soft sets. In their generalisation of fuzzy soft set, a degree is attached with the parametrization of fuzzy sets while defining a fuzzy soft set.

Let F_ρ and G_δ be two General Fuzzy Soft Set (GFSS) over the parametrized universe (U, E) . Hence, the $F_\rho = \{(F(e_i), \rho(e_i)), i=1,2,\dots,m\}$ and $G_\delta = \{(G(e_i), \delta(e_i)), i=1,2,\dots,m\}$. Let $\tilde{F} = \{F(e_i), i=1,2,\dots,m\}$ and $\tilde{G} = \{G(e_i), i=1,2,\dots,m\}$ are two families of fuzzy soft sets. Thus, the similarity between the two GFSS, F_ρ and G_δ is denoted as

$$S(F_\rho, G_\delta) = M(\tilde{F}, \tilde{G}) \bullet m(\rho, \delta). \text{ Here } M(\tilde{F}, \tilde{G}) = \max_i M_i(\tilde{F}, \tilde{G})$$

where,

$$M_i(\tilde{F}, \tilde{G}) = 1 - \frac{\sum_{j=1}^n |\tilde{F}_{ij} - \tilde{G}_{ij}|}{\sum_{j=1}^n (\tilde{F}_{ij} + \tilde{G}_{ij})} \tag{3}$$

is the similarity between \tilde{F} and \tilde{G} , and

$$m(\rho, \delta) = 1 - \frac{\sum |\rho_i - \delta_i|}{\sum (\rho_i + \delta_i)}, \text{ where } \rho_i = \rho(e_i) \text{ and } \delta_i = \delta(e_i). \tag{4}$$

is the similarity between two fuzzy sets ρ and δ .

if we used universal fuzzy soft set then $\rho = \delta = 1$ and $m(\rho, \delta) = 1$, and thus, the formula for similarity is

$$S(F_\rho, G_\delta) = M_i(\tilde{F}, \tilde{G}) = 1 - \frac{\sum_{j=1}^n |\tilde{F}_{ij} - \tilde{G}_{ij}|}{\sum_{j=1}^n (\tilde{F}_{ij} + \tilde{G}_{ij})} \tag{5}$$

where $\tilde{F}_{ij} = \mu_{\tilde{F}(e_i)}(x_j)$ and $\tilde{G}_{ij} = \mu_{\tilde{G}(e_i)}(x_j)$

Unfortunately, this similarity formula is less efficient when applied to large dataset and dataset is sparse. This formula will produce non-sparse element in large numbers and cause the calculation process to be slow. In the case of text document classification, usually size of dataset is very large and is very sparse, so the formula is not appropriate for the case of text classification, instead we will use a fuzzy similarity formula from [15]:

$$S(F_\rho, G_\delta) = \frac{\sum_{j=1}^n \min(\tilde{F}_{ij}, \tilde{G}_{ij})}{\sum_{j=1}^n \max(\tilde{F}_{ij}, \tilde{G}_{ij})} \quad (6)$$

This formula will be used to measure the similarity between two documents or in other words, measure the distance between two documents.

5 Fuzzy Soft Set Algorithm for Text Categorization

In this section we will discuss some of the algorithms based on fuzzy soft set theory to categorize text documents, so called Fuzzy Soft Set Classifier (FSSC). This algorithm consists of three important phases. First, pre-processing phase. Second, the training phase, and the third phase is the determination of the category or classification phase. The complete algorithm is as follows,

Pre-processing phase

1. Compute the *boolean*, *tf*, or *tf-idf* for all documents in the data set.
2. Compute fuzzification from step-1, using normalization.

Training phase

1. Given N documents obtained from the data class w .
2. Calculate the cluster center vector E_w using Equation-10

$$E_w = \frac{1}{N} \sum_{i=1}^N E_{wi} \quad (7)$$

3. Obtain a fuzzy soft set model for class w , where (\tilde{F}_w, E) , is a cluster center vector for class w having D features.
4. Repeat step 1-3 for all W classes.

Classification phase

1. Obtain a feature vector E_d from unknow class document.
2. Obtain the degree of membership, μ_{dw} , where $d = 1, 2, \dots, D$ and $w = 1, 2, \dots, W$ is calculated using Equation-11.

$$\mu_{dw} = \left[\sum_{j=1}^W \left(\frac{D_{dw}}{D_{dj}} \right)^{\frac{2}{m-1}} \right]^{-1} \quad (8)$$

where

$$D_{dw} = 1 - S(d, w) = 1 - \frac{\sum_{i=1}^N \min(E_{di}, C_{wi})}{\sum_{i=1}^N \max(E_{di}, C_{wi})}$$

$m = 2$ (fuzziness)

E_{di} : feature i^{th} of the unknown class document d

C_{wi} : feature i^{th} of the cluster center w

3. Assign the unknown class document to class w if

$$w = \arg \left(\max_{w=1}^W [\mu] \right) > \text{threshold}_w \quad (9)$$

the value of *threshold* can be set in accordance with the purposes.

6 Experiment Results

Experiments are performed with 70% training data set and 30% test data set are randomly selected, but we guarantee that the five classifier using a portion of the same data, and classification is done by using one-against-all, so the classifier will mark any documents included in the class of documents which are tested. For example, testing was conducted to document the class A and class rather than A. Then the classifier will provide the sign '1' on the documents included in the document class A, and gives a sign '-1' for documents that are not included in the document class A. Experiments are carried out to compare between five classifier, FSSC, kNN, Bayesian, Rocchio, and SVM. For bayesian classifier we used smoothing streng 0.01. while for the SVM software used SVMLight [27], with a linear kernel type and set the value of $C = 0.1$. All experiment performed on a 2.1 GHz Core 2 Duo computer with 2 GB memory using Octave 3.2.4.

6.1 Document Datasets

In our experiments, we used the Reuters-21578 document collection, which is considered as the standard benchmark for automatic document categorization systems [28]. The documents in Reuters-21578 have been collected from Reuters newswire in 1987. This corpus consists of 21,578 documents. 135 different categories have been assigned to the documents. The maximum number of categories assigned to a document is 14 and the mean is 1.24. This dataset is highly skewed. For instance, the earnings

category is assigned to 2,709 training documents, but 75 categories are assigned to less than 10 training documents. 21 categories are not assigned to any training documents. 7 categories contain only one training document and many categories overlap with each other such as grain, wheat, and corn. We selected 12 class that has the highest number of document.

6.2 Evaluation Metrics

To evaluate the performance of the soft set classifier we use the commonly used F-measure metric, which is equal to the harmonic mean of recall (ρ) and precision (π) [10]. ρ and π are defined as follows

$$\pi_i = \frac{TP_i}{TP_i + FP_i}; \rho_i = \frac{TP_i}{TP_i + FN_i} \quad (10)$$

Here, TP_i (True Positives) is the number of documents assigned correctly to class i ; FP_i (False Positives) is the number of documents that do not belong to class i but are assigned to class i incorrectly by the classifier; and FN_i (False Negatives) is the number of documents that are not assigned to class i by the classifier but which actually belong to class i .

The F-measure values are in the interval (0,1) and larger F-measure values correspond to higher classification quality.

Macro-averaged F-Measure. In macro-averaging, F-measure is computed locally over each class first and then the average over all classes is taken. and are computed for each class as in Equation-13. Then F-measure for each category i is computed and the macro-averaged F-measure is obtained by taking the average of F-measure values for each category as:

Macro-averaged F-measure gives equal weight to each data and is therefore considered as an average over all the class pairs. It tends to be dominated by the classifiers performance on common classes.

$$F(\text{macro} - \text{averaged}) = \frac{\sum_{i=1}^M F_i}{M} \quad (11)$$

$$\text{Where, } F_i = \frac{2\pi_i\rho_i}{\pi_i + \rho_i}$$

M is total number of classes. Macro-averaged F-measure gives equal weight to each class, regardless of its frequency. It is influenced more by the classifier's performance on rare classes.

6.3 Results and Discussion

The Table 1 shown the results of performance comparison test of five classifier, namely FSSC, kNN, Bayesian, Rocchio, and SVM. Testing performed for three types of document representation which is in the form of Term Frequency (TF), Term Frequency – Invert Document Frequency (TF-IDF), and Boolean.

We perform five types of measurements, namely precision, recall, f-measure, running time, and return size. Precision to measure the number of documents marked ‘1’ (or retrieved) correctly relative to the number of all documents marked as ‘1’, precision is ideal if all documents are marked ‘1’ represents all the relevant documents contained in the collection, and no irrelevant documents marked ‘1’. Recall to measure the number of documents marked as ‘1’ correctly, relative to the number of relevant documents, the number of all documents belonging to the same class in the collection. Ideal recall, if all documents are marked as ‘1’ consists of all relevant documents in the collection. F-measure is a combination of precision and recall, by a factor of a certain balance. This factor can be set by the user, typically used 0.5 as value of this balance factor.

Return size is the number of documents marked as ‘1’ by the classifier relative to total documents in collection. Ideally return size equals the number of relevant documents in the collection. Running time is the amount of time required to perform the classification is calculated in units of seconds.

Table 1. Performance comparison between five classifiers

PRECISION					
Doc	FSSC	kNN	Bayesian	Rocchio	SVM
TF	55.3%	64.3%	33.0%	26.8%	81.7%
TF-IDF	59.4%	64.7%	33.0%	35.6%	73.8%
Boolean	53.4%	65.2%	33.0%	25.8%	82.3%
average	56.0%	64.7%	33.0%	29.4%	79.3%

RECALL					
Doc	FSSC	kNN	Bayesian	Rocchio	SVM
TF	58.6%	71.2%	89.2%	90.1%	72.3%
TF-IDF	66.3%	68.3%	89.2%	93.5%	74.5%
Boolean	59.1%	68.5%	89.2%	92.7%	72.5%
average	61.3%	69.3%	89.2%	92.1%	73.1%

F-MEASURE					
Doc	FSSC	kNN	Bayesian	Rocchio	SVM
TF	56.5%	67.3%	44.6%	38.0%	76.6%
TF-IDF	62.3%	66.4%	44.6%	48.9%	74.0%
Boolean	55.9%	66.6%	44.6%	37.0%	76.8%
average	58.2%	66.7%	44.6%	41.3%	75.8%

Return size (Relevan size = 4.2%)					
Doc	FSSC	kNN	Bayesian	Rocchio	SVM
TF	4.5%	4.4%	10.1%	11.7%	3.9%
TF-IDF	4.4%	4.4%	10.1%	8.8%	4.2%
Boolean	4.4%	4.4%	10.1%	12.2%	3.9%
average	4.4%	4.4%	10.1%	10.9%	4.0%

Running Time (seconds)					
Doc	FSSC	kNN	Bayesian	Rocchio	SVM
TF	5.74	236.17	0.76	0.50	29.11
TF-IDF	5.88	255.94	0.75	0.53	29.77
Boolean	5.69	229.47	0.76	0.48	23.19
average	5.77	240.53	0.75	0.50	27.36

Table 1 shows that in general there is no absolute winner in this comparison, each has advantages and disadvantages. SVM classifier has a return-size (4.0%) the smallest among the other, even smaller than the average size of the relevant documents in the collection (4.2%). SVM also has the best precision for all types of document representations. The highest precision is achieved for a boolean representation of 82%. SVM also has the highest f-measure, 75%. However, SVM worked relatively longer (27.36 seconds) than the three other classifier, SVM only faster than kNN. kNN and Bayesian have performance that is almost not affected by those type of document representation, kNN has a uniform value of performance, precision (64.7%), recall (69.3%), and f-measure (66.7%) and has a return-size is slightly larger than the number of relevant documents in the collection (4.4%) rate is equal to return-size of the FSSC. Based on these four performance, kNN looks pretty good just slightly below the SVM, however, kNN works very slowly (240.53 seconds), tens or even hundreds of times slower than the other classifier.

Bayesian have a high recall (89.2%) but at the same time have low precision (33.0%), so the f-measure bayesian has a relatively low (44.6%), only slightly above Rocchio. High recall performance on Bayesian classifier, is influenced also by the return-size that relatively large (10.1%) was almost two times larger than the average size of relevant documents in the collection. However Bayesian work relatively quickly, less than a second (0.75 second), this speed is almost 30 times faster than SVM.

Rocchio classifier has a performance similar to Bayesian, can work even faster, in just 0.5 seconds, quicker than the four other classifier. Rocchio also has the highest recall (92.1%), it is also influenced by the high return-size (10.9%) even the highest among the four other classifier. However Rocchio classifier has the lowest precision (29.4%), among the four classifier and Rocchio also has the lowest f-measure (41.3%).

The FSSC classifier in general have a relatively good performance than any others classifier. Works relatively fast (5.77 second) than SVM and kNN, has the same return-size with kNN (4.4%), only slightly above the average number of relevant documents (4.2%). The FSSC has a precision (56%), recall (61.3%), and f-measure (58.2) slightly below the SVM and kNN, when compared with the Bayesian and Rocchio, FSSC has a precision and f-measure are relatively better. Figure-1 through figure-5, shows the results of the comparison in graphical form.

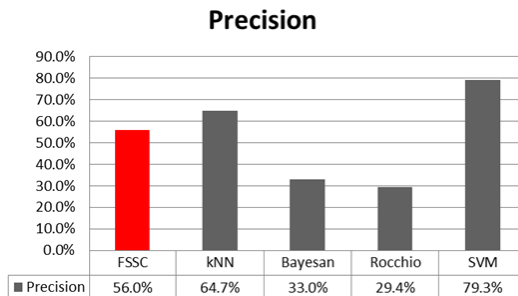


Fig. 1. The result of precision comparison

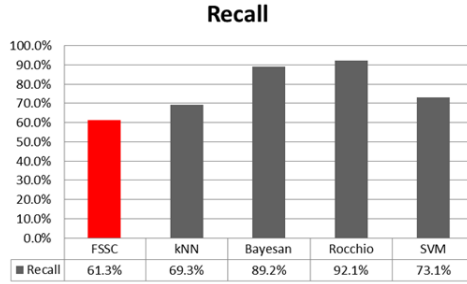


Fig. 2. The result of recall comparison

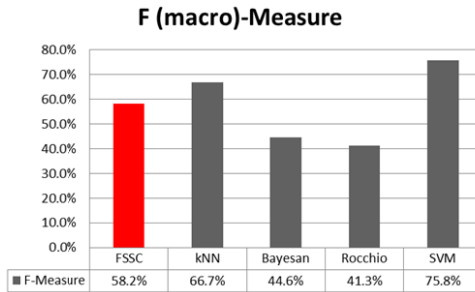


Fig. 3. The result of f-measure comparison

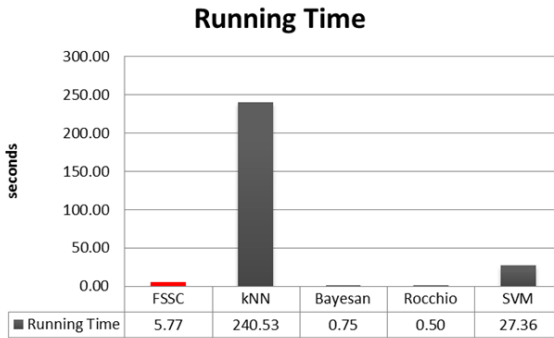


Fig. 4. The result of running time comparison

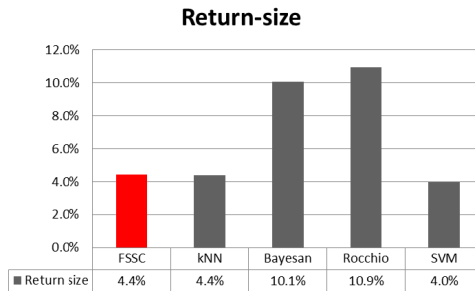


Fig. 5. The result of return-size comparison

7 Conclusion and Future Work

In this paper we investigate a new algorithm for text categorization based on fuzzy soft set theory, so called FSSC. We use the standard Reuters-21578 dataset and tf, tf-idf, and boolean weighting schemes to compare FSSC with four other types of text classifier consisting of kNN, Bayesian, Rocchio, and SVM. Classification proses is done by one-against-all, evaluation is done by using five types of measures, namely precision, recall, F-measure, return-size, and running time. In general there is no absolute winner in this comparison. Highest precision (79.3%) and F-measure (75.8%) was achieved by the SVM, while the recall (92.1%) the highest produced by Rocchio. Running time fastest was obtained by Rocchio too (0.5 second). Nevertheless Rocchio have low precision and low F-measure. The kNN has the precision, recall, and F-measure is relatively good but it works very slowly. Bayesian can work faster but similar to Rocchio, have low precision and low F-measure. The FSSC has precision, recall, and F-measure lower than SVM, and kNN but FSSC can work faster than both. When compared with the Bayesian and Rocchio, the FSSC works more slowly but has a higher precision and F-measure.

From the results above FSSC has a very good chance to be applied in information retrieval systems. For future research we will apply the FSSC on a search engine.

Acknowledgments. This work was supported by UTHM under the FRGS Grant No. 0736.

References

- [1] Sebastiani, F.: Machine learning in automated text categorization. *ACM Comput. Surv.* 34(1), 1–47 (2002)
- [2] Özgür, A., Özgür, L., Güngör, T.: Text Categorization with Class-Based and Corpus-Based Keyword Selection. In: Yolum, p., Güngör, T., Gürgen, F., Özturan, C. (eds.) *ISCIS 2005. LNCS*, vol. 3733, pp. 606–615. Springer, Heidelberg (2005)
- [3] Ozgur, L., Gungor, T.: Text classification with the support of pruned dependency patterns. *Pattern Recogn. Lett.* 31, 1598–1607 (2010)
- [4] Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features, pp. 137–142
- [5] Ozgur, L., Gungor, T., Gurgen, F.: Adaptive anti-spam filtering for agglutinative languages: a special case for Turkish. *Pattern Recogn. Lett.* 25, 1819–1831 (2004)
- [6] Yin, S., Wan, B., Qiu, Y., et al.: A Web Text Fuzzy Classification Algorithm on Fuzzy Comprehensive Weighted Evaluation Reasoning, pp. 1114–1117
- [7] Szmids, E., Kacprzyk, J.: Using Intuitionistic Fuzzy Sets in Text Categorization. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) *ICAISC 2008. LNCS (LNAI)*, vol. 5097, pp. 351–362. Springer, Heidelberg (2008)
- [8] Duan, Q., Miao, D., Chen, M.: Web Document Classification Based on Rough Set. In: An, A., Stefanowski, J., Ramanna, S., Butz, C.J., Pedrycz, W., Wang, G. (eds.) *RSFDGrC 2007. LNCS (LNAI)*, vol. 4482, pp. 240–247. Springer, Heidelberg (2007)

- [9] McCallum, A., Nigam, K.: A Comparison of Event Models for Nave Bayes Text Classification. In: Sahami, M. (ed.) Proc. of AAAI Workshop on Learning for Text Categorization, Madison, pp. 41–48 (1998)
- [10] Yang, Y., Liu, X.: A re-examination of text categorization methods. In: SIGIR 1999, pp. 42–49 (1999)
- [11] Mushrif, M.M., Sengupta, S., Ray, A.K.: Texture Classification Using a Novel, Soft-Set Theory Based Classification Algorithm. In: Narayanan, P.J., Nayar, S.K., Shum, H.-Y. (eds.) ACCV 2006. LNCS, vol. 3851, pp. 246–254. Springer, Heidelberg (2006)
- [12] Feng, F., Jun, Y.B., Liu, X., et al.: An adjustable approach to fuzzy soft set based decision making. *Journal of Computational and Applied Mathematics* 234(1), 10–20 (2010)
- [13] Roy, A.R., Maji, P.K.: A fuzzy soft set theoretic approach to decision making problems. *Journal of Computational and Applied Mathematics* 203(2), 412–418 (2007)
- [14] Bezdek, J.C., Ehrlich, R., Full, W.: FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences* 10(2-3), 191–203 (1984)
- [15] Widiantoro, D.H., Yen, J.: A fuzzy similarity approach in text classification task, vol. 2, pp. 653–658
- [16] Manning, C.D., Raghavan, P., Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press (2008)
- [17] Salton, G., Yang, C., Wong, A.: A Vector-Space Model for Automatic Indexing. *Communications of the ACM* 18(11), 613–620 (1975)
- [18] Lodhi, H., Saunders, C., Shawe-Taylor, J., et al.: Text classification using string kernels. *J. Mach. Learn. Res.* 2, 419–444 (2002)
- [19] Forman, G.: An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.* 3, 1289–1305 (2003)
- [20] ftp (2010), <ftp://ftp.cs.cornell.edu/pub/smart>
- [21] Porter, M.F.: *An algorithm for suffix stripping*, pp. 313–316. Morgan Kaufmann Publishers Inc., San Francisco (1997)
- [22] Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval, pp. 513–523
- [23] Maji, P.K., Biswas, R., Roy, A.R.: Soft set theory. *Computers & Mathematics with Applications* 45(4-5), 555–562 (2003)
- [24] Molodtsov, D.: Soft set theory—First results. *Computers & Mathematics with Applications* 37(4-5), 19–31 (1999)
- [25] Maji, P., Biswas, R., Roy, A.: Fuzzy soft sets. *J. Fuzzy Math.* 9(3), 589–602 (2001)
- [26] Majumdar, P., Samanta, S.K.: Generalised fuzzy soft sets. *Computers & Mathematics with Applications* 59(4), 1425–1432 (2010)
- [27] Joachims, T.: Making large-scale support vector machine learning practical. In: *Advances in Kernel Methods*, pp. 169–184. MIT Press (1999)
- [28] Lewis, D.D.: *Reuters-21578 Document Corpus V1.0* (2011)

Cluster Size Determination Using JPEG Files

Nurul Azma Abdullah, Rosziati Ibrahim, and Kamaruddin Malik Mohamad

Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia, Johor, Malaysia
{azma,rosziati,malik}@uthm.edu.my

Abstract. File can be recovered by simply using traditional recovery means. However, a technique is required to distinguish one file to another when dealing with hard disk with corrupted filesystem metadata. As in a computer file system, a cluster is the smallest allocation of disk space to hold a file, information about the cluster size can help in determining the start of file which can be used to distinguish one file to another. This paper introduces a method for acquiring the cluster size by using data sets from DFRWS 2006 and DFRWS 2007. A tool called PredClus is developed to automatically display the predicted cluster size according to probabilistic percentage. By using PredClus, the cluster size used in both DFRWS 2006 and DFRWS 2007 can be determined. Thus, JPEG images that are not located at the starting address of any cluster are most probably thumbnails or embedded files.

Keywords: Cluster size, JPEG files, File carving, DFRWS 2006/2007.

1 Introduction

In a computer file system, a cluster or in DOS 4.0 known as an allocation unit or in UNIX System as block, is the smallest logical set that is created to perform actual erasure for files and directories [1], [2]. A cluster is a contiguous group of sectors that contains an identical amount of data [3]. The grouping of sectors into clusters is performed by the operating system and thus is not a physical delimitation. Every track on the disk consists of the same number of clusters, and every cluster consists of the same number of sectors. A cluster usually composed of one to sixty four sectors with the amount of data for each sector is 512 bytes [3]. For a disk that uses 512-byte sector, a 512-byte cluster constitutes one sector, whereas a 4kb cluster constitutes eight sectors. The typical size of cluster varies from 1 sector to 128 sectors which is equivalent to 64kb [2], [4].

A cluster is the smallest allocation of disk space to hold a file [5], [6]. Files in the computer are stored in one or more clusters on the disk. A cluster may contain the whole file or portion of files but a cluster only stores data for one file [3], [7]. Hence, there will be several sectors available that not been used to store other data even the file does not contain enough data to fill all the sectors of the cluster. Because of this, it is an issue for the storage industry whether to allow larger cluster sizes or not. There are advantages and disadvantages for both options. To store small files on a file

system with large clusters will waste disk space which is called slack space [5], [8], [10]. However, a large cluster size reduces bookkeeping and fragmented files. This may improve reading and writing speed [1], [5]. Overall, no matter what is the size of cluster, it is safe to say that the start of one file must be at the beginning of cluster. Therefore, it is important to determine the cluster's size to determine the start of file. This information is useful for both steganography and file carving.

In file carving, the whole content of the hard disk used for evidence need to be imaged (will be referred to as image file) in preparation for forensics investigation. The image file contains thousands of hex code representing all files in the hard disk used for evidence. It is impossible to read line by line manually in order to extract all files into their original form. Information about the start of file can be used to identify each file that resides in any dataset.

Information of the start of file is also useful to determine slack space. This area is used by one of the steganography techniques for hiding message. The slack space size can be determined by subtracting the size of file from the used cluster size. In this paper, experiments are done using datasets from DFRWS.

DFRWS is started in 2001 as a non-profit, volunteer organization that is dedicated to the sharing of knowledge and ideas about digital forensics research. Datasets from DFRWS 2006 and 2007 challenges [10], [11] are used in the experiment because both datasets have been widely used in many file carving research such as [12- 18].

DFRWS 2006 and 2007 challenge datasets are prepared using an assumed 512-byte sector size. However, it is important to know the cluster size for the dataset as it is the smallest unit used by the computer system for allocating data. This paper focuses on finding the cluster size for data set of DFRWS 2006 and DFRWS 2007 challenge. The proposed technique, PredClus is developed to automatically display the predicted cluster size according to probabilistic percentage. This information can be used as an alternative method in recognizing thumbnail and embedded files.

The remainder of this paper is structured as follows: in the next section is discussing the proposed technique, PredClus and the experimentation done. Section 3 presents experimental results and the discussion. The conclusion is presented in Section 4.

2 PredClus

Formally, the size of a cluster is determined by the size of the volume depending on the type of file system such as NTFS, FAT16, FAT 32 or ExFAT. Table 1 shows the default cluster size for any operating system that supports FAT [6]. The cluster size for DFRWS 2006 and 2007 datasets are not following the default cluster size as stated in Table 1. There is no available information about the cluster size but the sector size for both datasets is assumed to be 512 bytes. For this experiment, JPEG files are used to determine the cluster size for these datasets. These JPEG images are recognized from their standard header with additional validated header [12], [13] for different types of JPEG. The information for each JPEG file location or offset can be obtained from the DFRWS website [10-11]. However, embedded JPEG images are not

included in the layout. Therefore, standard headers of JPEG images are searched from the data set by using the hex code header pattern. The header will be paired with suitable JPEG end of file for retrieving the whole file. With the help of the file layout prepared in DFRWS, we then reassemble some files that are fragmented manually for the purpose of this experiment. Then, we checked for each file retrieved and only files that can be viewed in image editor are used for the experiment.

Table 1. Default cluster size for FAT compatible OS

Drive size (logical volume)	FAT Type	Sectors	Cluster size
15MB OR less	12 bit	8	4kb
16MB -127MB	16 bit	4	2kb
128MB – 255MB	16 bit	8	4kb
256MB – 511MB	16 bit	16	8kb
512MB–1023MB	16 bit	32	16kb
1024MB-2048MB	16 bit	64	32kb
2048MB-4096MB	16 bit	128	64kb
4096MB-8192MB	16 bit	256	128kb
8192MB-16384MB	16 bit	512	256kb

According to [19-20], 512 bytes is the common sector size. Thus, the size for a cluster that contain only one sector is 512 bytes. The maximum cluster size for any operating system that supports FAT is 256 kb. Nevertheless, Microsoft only use cluster size of up to 64 kb in an effort to overcome the weakness of their FAT File systems. Although bigger size cluster can reduce disk accesses and make faster

performance, it can also induce space issues. In a condition where a computer system stores many small files, this will result in many slack spaces (bigger wasted space). This is because of the characteristic of cluster unit that allows only one file per cluster. In this case, it will waste computer storage and reduce its efficiency. DFRWS 2006 and DFRWS 2007 datasets are 50Mb and 330Mb respectively. The maximum cluster size tested is up to 8kb. This size is accommodating as recommended in [7].

2.1 Proposed Architecture

PredClus system architecture is illustrated in Fig. 1. Based on Fig. 1, PredClus is used to predict cluster size for dataset used in the digital forensics investigation. In this research, DFRWS 2006 and 2007 dataset are used as data images. PredClus then processes the data image and finally generates a report on the cluster size used in the data image.

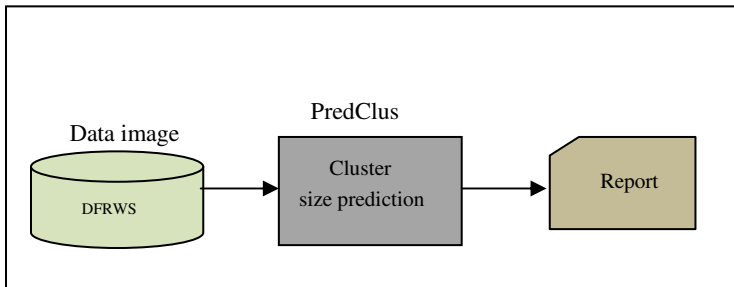


Fig. 1. PredClus System Architecture

2.2 Proposed Algorithm

In this section, an algorithm (illustrated in Fig. 2) is introduced to predict cluster size used in both DFRWS 2006 and DFRWS 2007.

First, data from dataset is read. These data are in hex values. The hex values then matched with the standard JPEG header. However, in this experiment, additional markers are also used instead of standard JPEG header, 0xFFD8 alone. The additional markers used are 0xFFE0, 0xFFE1, 0xFFE2, 0xFFC4 and 0xFFDB. When matched, the offset for each marker matched is retrieved. Using formula as mentioned earlier, the determinant value is calculated. If the determinant value = 0, then file found is counted. This is done for each cluster size which are 512-byte, 1-kb, 2-kb, 4-kb and 8-kb cluster as shown in Fig. 3 and Fig. 4.

After the determinant value for all JPEG files in the datasets is extracted, files found for each cluster size then are summed. The percentage for each cluster size is calculated. Then, a report is produced.

1. Read data image
2. Initialize ClustSize[] = {512, 1024 (1kb), 2048 (2kb), 4096 (4kb), 8192 (8kb)}
3. Initialize Det_value
4. Initialize Counter[]={0, 0, 0, 0, 0, 0} // to store the number of headers found for ClustSize[]
5. Initialize Counter[]={0, 0, 0, 0, 0, 0} // to store the number of headers not at start of ClustSize[]
6. Initialize $i = 0$ // used for ClustSize index
7. Initialize $k = 0$ // used for Counter index
8. Find JPEG header
9. If found
 10. Read CurrentOffset
 11. Det_value = CurrentOffset % ClustSize[i]
 12. If Det_value ==0
 13. increment Counter[k] by 1
 14. Else
 15. increment noneCounter[k] by 1
16. if not end of data image, repeat step 7
17. Increment i by 1
18. Increment k by 1
19. If $i \leq 4$, repeat step 7
20. Generate report

Fig. 2. Algorithm used in PredClus for predicting data image cluster size

Offset	Mod_512	Mod_1024	Mod_2048	Mod_4096	Mod_8192
240357	229	741	741	2789	2789
1980416	0	0	0	2048	6144
1980748	332	332	332	2380	6476
1995443	179	691	691	691	4787
4241920	0	512	512	2560	6656
5948928	0	512	1536	1536	1536
5949358	430	942	1966	1966	1966
6257664	0	0	1024	3072	7168
14134784	0	512	1536	3584	3584
16115200	0	512	1536	1536	1536
16144896	0	512	512	2560	6656
18581504	0	0	0	2048	2048
20806656	0	0	1024	3072	7168
21304832	0	512	1536	1536	5632
22238208	0	0	1024	1024	5120
23329792	0	0	1024	3072	7168
23330000	208	208	1232	3280	7376
24017920	0	0	1024	3072	7168
48561152	0	0	1024	3072	7168
48561270	306	818	1842	3890	7286

Fig. 3. Example of output from formula used in experiment using data from DFRWS 2006

Offset	Mod_512	Mod_1024	Mod_2048	Mod_4096	Mod_8192
4763250	114	626	1650	3698	3698
9789897	457	457	457	457	457
13998051	483	995	2019	2019	6115
18892751	463	975	1999	1999	1999
19014836	180	180	1204	1204	1204
19623764	340	852	1876	3924	3924
23316537	57	57	57	2105	2105
23548038	134	134	134	134	4230
29258240	0	512	512	512	4608
29260288	0	512	512	2560	6656
29263360	0	512	1536	1536	1536
35857408	0	0	1024	1024	1024
35859568	112	112	1136	3184	3184
35890554	378	378	1402	1402	1402
39887467	107	619	619	619	619
40613451	75	587	1611	1611	5707
42441305	89	601	601	2649	6745
44910592	0	0	0	2048	2048
45582848	0	512	512	2560	2560
45587968	0	512	1536	3584	7680
45593088	0	512	512	512	4608
46273024	0	512	512	512	4608
46274430	382	894	1918	1918	6014
48015360	0	0	0	2048	2048
48020480	0	0	1024	3072	7168
48026112	0	512	512	512	4608
58263114	74	586	1610	1610	1610

Fig. 4. Example of output from formula used in experiment using data from DFRWS 2007

2.3 Experimentation

Generally, a cluster size is represented by the data section size of sectors in a cluster multiplied by the number of sectors [20]. Hence, a cluster size can be derived from the formula below:

$$\text{Cluster_size} = \text{size_of_cluster} \times \text{number_of_clusters}$$

For this experiment, as valid cluster size values are powers of two with at least 256 and at most 65536 bytes per cluster [4], we choose to use cluster of 512 bytes, 1kb, 2kb, 4 kb and 8kb. Hence, the cluster size for each group using the formula above is as listed in Table 2.

Table 2. Values (in bytes) for each cluster group

Cluster group	Value of each cluster group
512byte	512 x 1 = 512
1kb	512 x 2 =1024
2kb	512 x 4 =2048
4kb	512 x 8 =4096
8kb	512 x 16 =8192

Next, all offset for each JPEG files are derived using a simple program to detect JPEG header and then manually checked, reassembled and validated for completeness. The example of offsets for JPEG files in DFRWS 2006 is as shown in Table 3. Then, using the formula listed below, we can determine which cluster size is used in DFRWS 2006 and DFRWS 2007.

$$\text{Determinant_value} = \text{Offset mod cluster_size_value}$$

The determinant value is derived from above formula where offset is the offset for the tested JPEG file and cluster size value is the cluster size for each cluster group 512 bytes, 1kb, 2kb, 4kb or 8kb. This determinant value can result in either 0 or greater than 0. The simple rule for this formula is, if the determinant value results in 0, it indicates that the header of the file is at the start of cluster. Otherwise, it does not at the start of cluster.

Table 3. Offsets prepared for each JPEG files in DFRWS 2006

Number	Offsets
1	240357
2	1980748
3	1995443
4	4241920
5	5948928
6	5949358
7	6257664
8	14134784
9	16115200
10	16144896
11	18581504
12	20806656
13	21304832
14	22238208
15	23329792
16	23330000
17	24017920
18	48561152
19	48561970

3 Result and Discussion

The result of the experiments can be clearly examined in Figure 5 and Figure 6. These data then are illustrated in a form of bar charts as shown in Figure 7 and Figure 8.

From the figures (Figure 5, Figure 6, Figure 7 and Figure 8), clearly we can see the highest percentage of files found is using 512-byte cluster size which is 9.86% for DFRWS 2006 dataset and 21.13% for DFRWS 2007 dataset. This is equal to 14 files found for DFRWS 2006 and 30 files found for DFRWS 2007. If we look at the last cluster size variable which is *none*, it is refers to any JPEG file offset that could not fit in any cluster group listed which is 6 and 112 files for DFRWS 2006 and DFRWS 2007 respectively. This is a very interesting finding. We know that these files do not at the start of cluster but still some of that files are valid and complete. We then, check these file and we find out that some of these files are thumbnails and embedded files. These kinds of file normally do not start at the start of cluster because they are part of a file. Thumbnails are reduced-size versions of pictures while embedded files refer to JPEG file that is embedded in other types of file such as Word, Excel, pdf, etc. Although in DFRWS 2007, the other cluster group fit for some JPEG files, but overall, we are confident that dataset for both DFRWS 2006 and 2007 use 512-byte cluster. In a conclusion, we find out with the assumption that the cluster size is 512 bytes, an original JPEG file always start at the start of cluster, while thumbnails and embedded JPEG files can start at any location within the original file.

Cluster size	Files detected	Not at start
512 byte	14	6
1kb	8	12
2kb	2	18
4kb	0	20
8kb	0	20
All	14	6

Cluster size	Percentage files detected
512 byte	9.86
1kb	5.63
2kb	1.41
4kb	0.00
8kb	0.00

Highest percentage for JPEG headers found at start of cluster for DFRWS 2006 is 9.85915%

Fig. 5. Statistics result from experiment using DFRWS 2006 dataset

Cluster size	Files detected	Not at start
512 byte	30	112
1kb	17	125
2kb	12	130
4kb	6	136
8kb	4	138
All	30	112

Cluster size	Percentage files detected
512 byte	21.13
1kb	11.97
2kb	8.45
4kb	4.23
8kb	2.82

Highest percentage for JPEG headers found at start of cluster for DFRWS 2007 is 21.1268%

Fig. 6. Statistics result from experiment using DFRWS 2007 dataset

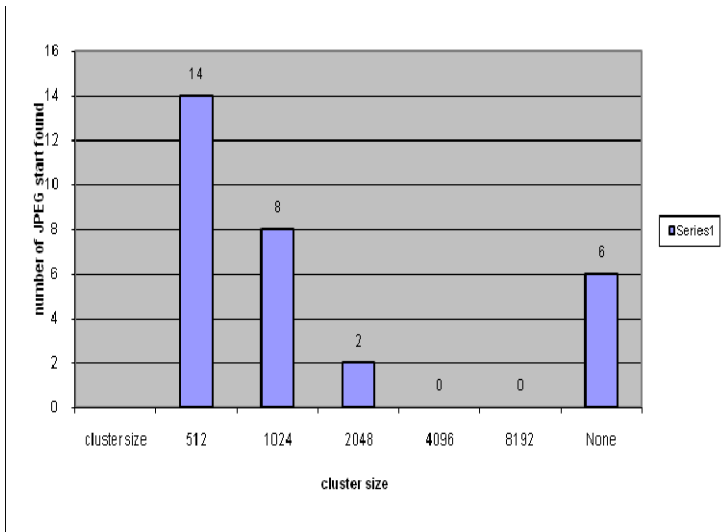


Fig. 7. Statistics for DFRWS 2006

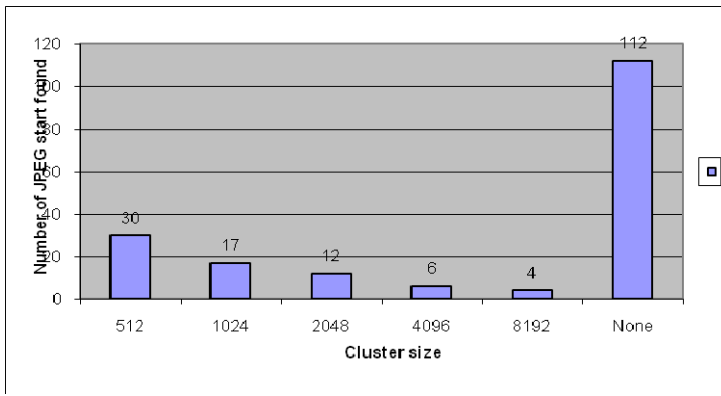


Fig. 8. Bar chart for DFRWS 2007

Through these finding, we suggest to use cluster size as part of file carving process to distinguish between the original file and thumbnails or embedded files. Header of file found not at the start of cluster suggests that the file is not original JPEG file but may be the thumbnail or embedded JPEG image. Hence, in carving contiguous files, we can say that a file found not at the start of cluster is not the original JPEG image while in carving fragmented files, this condition shows that the first file is not fragmented with another JPEG file but the file contains thumbnail for the original. Therefore, the second header indicates either a fragmentation point or a thumbnail for the file. Furthermore, the information about cluster size can be used to determine available slack space in a cluster.

4 Conclusion

A cluster is the smallest allocation of disk space to hold a file. Therefore, information about cluster size can help in determining the start of file as its characteristic that does not allow more than one file in a single cluster. This information can be used both in improving file carving and steganography technique. There are various techniques to carve file from corrupted hard disk either using pattern matching, statistical or other techniques. All of these techniques require the tool to distinguish one file to another. However, the files that reside in the disk image typically are shown as a single file. This means that all the files in the corrupted data are seen as one file in a disk image. This file contains thousand lines of hex code. It is impossible to determine the start of each file without thoroughly check the file and matched with all patterns available. Furthermore, from the experiment for both datasets, clearly we can see there are situations where the header is not at the start of file. Further investigation, we found out that these situations are because of the JPEG header belongs to either thumbnail or embedded file. We can say that if any JPEG header found not at the start of file, it indicates that file is either thumbnail or embedded file. Hence, information about cluster size do helping in distinguish original JPEG file with thumbnail/s and embedded image.

As in steganography, information about start of file can help to determine the slack space which is commonly used to hide messages. Hence, to know the start of file is useful for both file carving and steganography technique.

Acknowledgement. The authors would like to thank Universiti Tun Hussein Onn Malaysia (UTHM) for supporting this research.

References

1. Ng, S.W.: Advances in Disk Technology: Performance Issues. *Computer* 31, 75–81 (1998)
2. File Allocation Table, http://en.wikipedia.org/wiki/File_Allocation_Table#Boot_Sector
3. Jemigan, R.P., Quinn, S.D.: Two-Pass Defragmentation of Compressed Hard Disk Data with a Single Data Rewrite. U.S Patent 5574907
4. Mkfs.xfs(8)-Linux Man Page, <http://linux.die.net/man/8/mkfs.xfs>
5. Data Cluster, http://en.wikipedia.org/wiki/Data_cluster
6. The Default Cluster Size for the NTFS and FAT File Systems, <http://support.microsoft.com/kb/314878>
7. Default Cluster Size for NTFS, FAT, and ExFAT, <http://support.microsoft.com/kb/140365>
8. Linux System Administrator Guide: Chapter 5: Using Disks and Other Storage Media, <http://tldp.org/LDP/sag/html/filesystems.html>
9. Vista/XP Install on Large Cluster Sizes, <http://www.winvistatips.com/vista-xp-install-large-cluster-sizes-t801412.html>
10. Digital Forensics Research Workshop (DFRWS), <http://www.dfrws.org/2006/challenge/submission.shtml>
11. Digital Forensics Research Workshop (DFRWS), <http://www.dfrws.org/2007/challenge/submission.shtml>
12. Mohamad, K.M., Mat Deris, M.: Single-byte-marker for Detecting JPEG JFIF Header using FIRIMAGE-JPEG. In: Proc. of the 2009 Fifth International Joint Conference on INC, IMS and IDC, 2009, pp. 1693–1698 (2009)
13. Mohamad, K.M., Herawan, T., Deris, M.M.: Dual-Byte-Marker Algorithm for Detecting JFIF Header. In: Bandyopadhyay, S.K., Adi, W., Kim, T.-h., Xiao, Y. (eds.) ISA 2010. CCIS, vol. 76, pp. 17–26. Springer, Heidelberg (2010)
14. Mohamad, K.M., Mat Deris, M.: Fragmentation Point Detection of JPEG Images at DHT Using Validator. In: Proc. of the 2009 FGIT, pp.173–180 (2009)
15. Mohamad, K.M., Patel, A., Herawan, T., Mat Deris, M.: myKarve: Jpeg Image And Thumbnail Carver. *Journal of Digital Forensic Practice* 3, 74–97 (2010)
16. Cohen, M.I.: Advanced Carving Techniques. *Digital Investigation* 4(1-4), 119–128 (2007)
17. Metz, J., Mora, R.J.: Analysis of 2006 DFRWS Forensic Carving Challenge, <http://sandbox.dfrws.org/2006/mora/dfrws2006.pdf>
18. Richard III, G.G., Roussev, V.: Scalpel: A Frugal, High Performance File Carver. In: Proc. of the 2005 Digital Forensics Research Workshop, New Orleans (2005)
19. McKusick, M.K., Joy, W.N., Leffler, S.J., Fabry, R.S.: A Fast File System for UNIX. *ACM Transactions on Computer Systems* 2 (1984)
20. Kanagawa, S.K.: Information Reproduction Apparatus and Information Reproduction Method. U.S. Patent 6,236,663 BI

Semantic Web Search Engine Using Ontology, Clustering and Personalization Techniques

Noryusliza Abdullah and Rosziati Ibrahim

Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia, Johor, Malaysia
{yusliza,rosziati}@uthm.edu.my

Abstract. Data accuracy and reliability have been a serious issue in the vast emergence of information on the web. Advanced web searching has assisted in knowledge retrieving. However, most knowledge on the Web is presented in natural-language text that understandable by human but difficult for computers to interpret. Therefore, Semantic Web approach is widely used to give more reliable application. This paper presents a framework in enhancing knowledge retrieval processes using Semantic Web technologies. Instead of using ontology and categorization alone, we are injecting personalization concept from Relational Database (RDB) to ensure more reliable data are obtained. The proposed framework is discussed in details. A case study is presented to see the viability of the proposed framework in retrieving the meaningful information.

Keywords: Semantic Web Search, Ontology, Clustering, User Profiling.

1 Introduction

Knowledge is so important for us in all aspects. Although many sources have given good numbers of information but there are still lacking in terms of knowledge reliability. As organization's ability to learn and handle knowledge processes or knowledge product is considered the new key success factor [1], research in information and knowledge retrieval are actively conducted. They are also useful in preventing researchers from digging every single document to information searching. However, retrieving the real meaning of data is often fails to give the desired result.

As Information Technology has evolved to the mature phase, we do not expect this situation should continue. Something has to be done to ensure we can learn from other people by capturing as much information or knowledge as we can and make it meaningful to our purpose. In order to make this determination is fulfilled, knowledge retrieval is chosen as an enabler to overcome the previous stated problem. It will be the next generation retrieval systems in order to overcome the rapid increase in data and information to find the right knowledge [2]. Nonetheless, in retrieving knowledge, substantial amount of efforts are needed. According to Tao, Li, & Nayak, [3] interpreting users' information needs is compulsory in knowledge retrieval. Hence, they proposed Local Instance Repository (LIR), a personal collection of web documents recently visited by the user.

The major challenge in implementing either information or knowledge retrieval in WWW is most knowledge on the Web is presented as natural-language text that understandable by human but difficult for computers to interpret. So, Semantic Web approach is widely used to give more reliable application. Mikroyannidis [4] explains that Semantic Web is able to give information a well-defined meaning and better cooperation between computers and people. In applying the Semantic Web, ontology is commonly discussed. It is an explicit specification of a conceptualization. dâ€™Aquin & Noy [5] states that data interoperability property from ontologies which permits sharing and reusing features, is a key promises of the Semantic Web. These advantages are highlighted in 11 ontologies libraries. Four of the libraries might take into consideration in this study due to the general domain. They are Cupboard[6], Ontology Design Patterns (ODP) [7], OntoSelect [8], OntoSearch2 [9] and Schema-Cache [10].

While ontologies are capable in giving good outcome, researchers are trying to enhance searching method using clustering technique [11] and user profiling/personalization [12, 13, 14]. Although previous researches are capable to give good results, we are motivated to improve the output. Therefore, we propose the hybrid of Semantic Web Search Engine, a knowledge retrieval platform using Semantic Web Search to ensure reliability criteria is fulfill in retrieving knowledge. This web searching based on three criteria: ontology, clustering and user profiling/personalization. The techniques are consolidated to give more reliable searching particularly in the user's perspective. The proposed technique will extract meaningful information and give positive impact in the area of Knowledge Retrieval.

The remainder of this paper is organized as follows. Section 2 lists related works that relevant to this research. Section 3 discusses the research method while Section 4 provides suggested framework in this study. Finally, section 5 provides the conclusion.

2 Related Works

In this section we discuss the details of Semantic Web, online ontology resources, clustering (or categorization) and user profiling (or personalization).

2.1 Semantic Web

In our research, we concentrate on the search engine. Semantic Web search engine rank semantic web document, RDF graphs, triples and terms. This is different from conventional search engines where only Web pages are ranked [15]. The functionality of the Semantic Web is resemble typical search engine such as Google and Yahoo but referring to Jiang [16] the benefit of using it is the ability for machine-understood descriptions of meaning. The web helps us to reach information that we search and other data related to it. Thus, Semantic Web is not just sharing text of a page but data and facts as well [17].

Other motivation to use Semantic Web is it helps in collecting data together from the web [17]. Referring to Mikroyannidis [4], Semantic Web is better than conventional web because of the ability to handle unstructured content. Semantic

Web can overcome this problem by using software agent that can enhancing search precision and enabling logical reasoning. Semantic Web is the significant product among the established companies like Oracle, Vodafone, Amazon, Adobe, Yahoo and Google wherein they provide a smarter web [18]. Moreover, Joo [19] views semantic web has a potential to implement semantic integration and reduce information overload.

According to Janev & Vrane [20] this is the popular area in the Information and Communication Technology field. Many research efforts are conducted to improve traditional web and making the content available on the semantic web. In line with this thought, Edwards [17] explains moving from HTML to XML is the original plan for the semantic web. Loopholes in HTML addressed by Linked Data that connect data, information and knowledge on the semantic web using Uniform Resource Identifier (URI) and Resource Description Framework (RDF).

2.2 Online Ontology Resources

Ontology is the heart of the Semantic Web. It is a domain and knowledge representation [21, 22]. In consonance with Hepp [23], ontologies are the vocabulary that can be used to express a knowledge base while Diez-Rodriguez *et al.* [24] discussed that the intention to represent concepts in ontologies is to improve knowledge searching and discovery mechanisms.

In-depth researches are conducted on ontologies because of the function as the backbone for the semantic web [20]. Joo [19] states that research on ontology is necessary to ensure the diffusion of the semantic web. In addition, Ontology-based knowledge organization can contribute to express the contents of information elements and semantic relations between them. It can also support semantic reasoning and retrieval [25]. Furthermore, Maier, Hadrich & Peinl [1] stated that documented knowledge which spread across multiple sources requires identification and visualization with the help of knowledge maps and integration supported by ontologies as a manager to semantic content.

However, in the interest of ensuring ontologies and metadata to represent information correctly, they need constant updates and maintenance [4]. In order to accomplish the aim, Web Ontology Language (OWL) is used. It is a semantic markup language for publishing and sharing ontologies on the World Wide Web and used to describe the classes and relations between them [21]. Still, according to Cardoso [18], building ontology is more complex in terms of logic and structure compared to building software. The main goal of ontology engineering is to produce useful, consensual, rich, current, complete, and interoperable ontologies.

In building ontologies, linking them to the knowledge organization systems is the main priority to increase interoperability and data accessibility [23]. The highest methodologies adoption in develop ontology is Methontology. Ontologies development needed an editor. There are several editors including Protégé, SWOOP, OntoEdit, OntoStudio and many more. Among all, protégé is the most used editor due to the support of wide variety of plugin and import formats and it's free open source.

In accordance with D'Amato *et al.* [26], combining semantic web search with ontological background is a promising research approach. New semantic web applications discover ontologies on the web. Exploring large-scale semantics need to perform certain tasks: Find relevant resources, Select appropriate knowledge, Exploit heterogeneous knowledge sources and combine ontologies and resources [27].

Semantic applications that use online knowledge can ensure in obtaining appropriate semantic resources. D'Aquin *et al.* [27] lists several Semantic web search engines such as Swoogle, Sindice, Falcon-S and Watson. Among these search engines, Watson is better in terms of finding, selecting, exploiting and combining online resources without having to download the ontologies. It uses a set of crawlers to explore sources to check for duplicates, copies or prior versions. Analyzing and indexing are depending on content, complexity, quality and relation to other resources.

2.3 Clustering/Categorization

Extension to the current approach, Trillo *et al.*[11] proposes categorization or clustering method which turns up with a semantic technique to group the output of searching keywords into different categories. They use online ontologies to define the possible categories.

2.4 User Profiling/Personalization

Research on personalization or user profiling in the semantic web is actively conducted. Jie *et al.* [12] uses information on the homepage for profile extraction. Data for instance, interest and publications are extracted to get more information on users. Other researchers are based on the history of visited site for personalization.

In order to improve browsing result, personalization mechanism is used. This mechanism is based on user preferences and monitoring process of user navigation. Antoniou *et al.* [13] suggests the method of suggesting highly accessed pages from the past users' navigational patterns to the new users. This method has overcome very frequent accessibility for short periods of time using advance data structures technique. Yoo [14] supports effective retrieval of personalized information on the semantic web by using hybrid query processing method. The hybrid of two methods, query rewriting method and reasoning method are able to process query when individual requirements change.

Many researchers are using user profiling and personalization term interchangeably and refer them as the same entity. However, some researchers adopt them as two different things. Personalization refers to the navigational behaviour while user profiling is user's personal data. We will use user profiling term from now onwards to avoid confusion. While most researchers are concentrating on browsing history and using web data for personalization or user profiling, we choose to hybrid our Semantic Web Search engine using data in our Relational Databases (RDB) to get more info on users. Due to the absence of Oracle-like RDMBS which implements RDF model to their databases, we map our RDB to the RDF.

3 The Framework of Semantic Web Search Engine

In this section, the proposed framework to implement hybrid Semantic Web Search is presented.

3.1 Semantic Web Search Construction

In retrieving knowledge, there are several techniques can be implemented. Semantic Web is chosen based on certain advantages stated in the previous section. Ensuring results obtained are more reliable, method in [11] is used with modification in user profiling concept.

3.2 Search Result Based on User Profiling

This research focuses on Universiti Tun Hussein Onn Malaysia (UTHM) dataset. Emphasizing on the user profiling, members' own data are extracted and used to ensure results are more reliable in user's perspective. These are the components need to be examined:

- Staff ID
- Staff Name
- Faculty ID
- Faculty Name

3.3 Proposed Model

Based on [11], the approach of extracting online ontology from the web is applied. The results are then categorized to facilitate users. However, searching facilities is optimized by adopting user profiling technique in the current approach. Figure 1 (b) shows the adaptation framework from Trillo *et al.* [11] (Refer Figure 1a).

Figure 1 shows the adaptation of Trillo *et al.* [11] with the enhancement in user profiling. Compared to the previous framework, this proposed framework will match categorized keywords with users' personal data and rank the output based on the data. In our approach, user's own data is compared to the clustered search result. Computer's name might be used for identification. Otherwise, users might key-in simple data for instance staff ID as recognition to give personalized result. Enabling the semantic search to drill data from the database, need particular method. It is due to the different RDF format used in the semantic web compared to the Relational Databases (RDB). RDF format is presented in subject, predicate, object format. Therefore, RDB to RDF mapping will be conducted. Referring to Matthias *et al.* [28], in application scenarios, Direct Mapping is more suitable in RDB to RDF cases. In this approach, relational tables are map to classes in RDF vocabulary and tables attributes to properties in the vocabulary. Hence, Direct Mapping is used for our framework in the user profiling part.

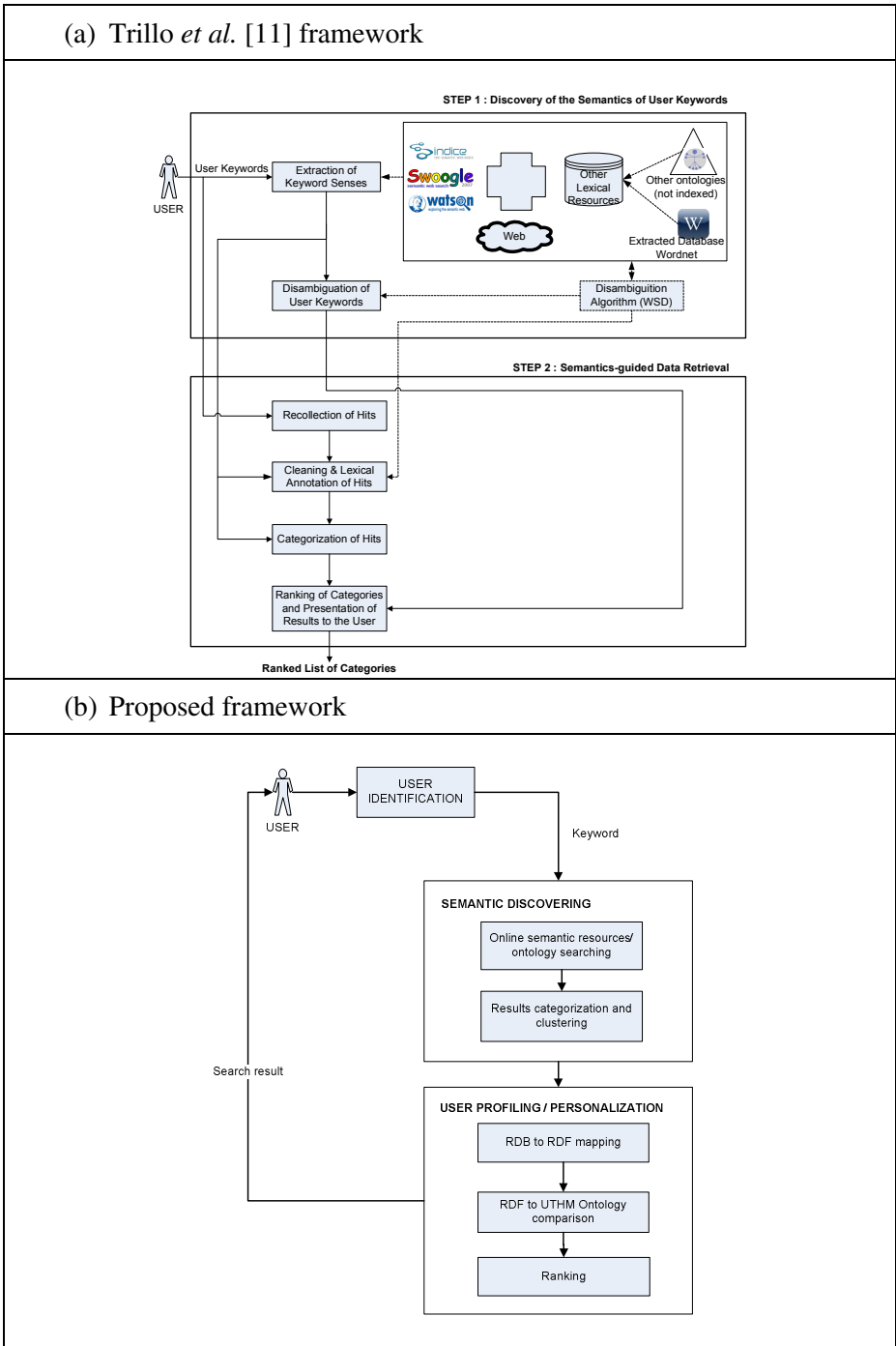


Fig. 1. (a) Trillo *et al.* [11] framework. (b) Proposed framework.

3.4 Algorithm

An Algorithm shown in Figure 2 is used in the framework. In line 1, users' entered ID as identification. The keyword entering, processing and categorizing are done in Line 2 to 4. In these steps, online ontology is used to specify and conceptualize the keywords. The main contribution of this research is between line 5 to 17. They utilized user profiling technique and rank the results. Combining these steps with online ontology and clustering is not implemented by Trillo *et al.* [11].

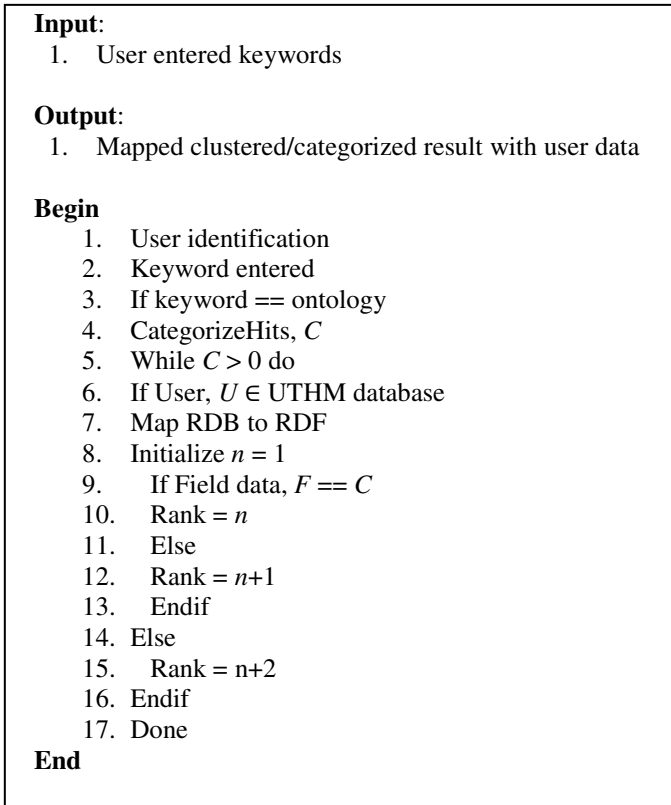


Fig. 2. Algorithm used in the Semantic web Search Engine

3.5 Data Description

Important attributes are listed in Table 1 and Table 2 with the structure and description. The data structure is based on real data from UTHM's Relational Database (RDB). In the implementation phases, actual UTHM data will be used as datasets.

Table 1. Datasets structure – Faculty Table

Field	Structure	Description
facID	Varchar2 (3)	Faculty in UTHM.
facName	Varchar2 (50)	Name of faculty.

Table 2. Datasets structure – Staff Table

Field	Structure	Description
staff ID	Varchar2 (10)	ID for every staff. Unique. Used as identification.
staffName	Varchar2 (50)	Name of staff.
facID	Varchar2 (3)	Faculty for staff.

4 Case Study of the Semantic Web Search Engine

This section describes a case study of Semantic Web Search Engine for UTHM members using three techniques: ontology online, clustering and user profiling. Algorithm in Figure 2 is explained in detail. Total of 968 academic staffs from UTHM are expected to utilize this finding. Table 3 shows academic staffs based on faculty. However, for testing requirement, only ten percent of them which selected randomly will undergo the testing phase.

Table 3. Academic staff

Faculty	Number of academic staff
Management	88
Civil/Environment	167
Mechanical	200
Electrical	201
Vocational	99
Information Technology (IT)	69
Science & Technology	144
TOTAL	968

* Data as of Wednesday, 18th January 2012

4.1 Step 1 - User Identification

The goal of this process is to capture user's profile. Figure 3 shows the Graphic User Interface (GUI) for identification. This search engine classify user's faculty. To

facilitate uses, computer's data stored in web log might be used to avoid users from enter ID every time they use this application.

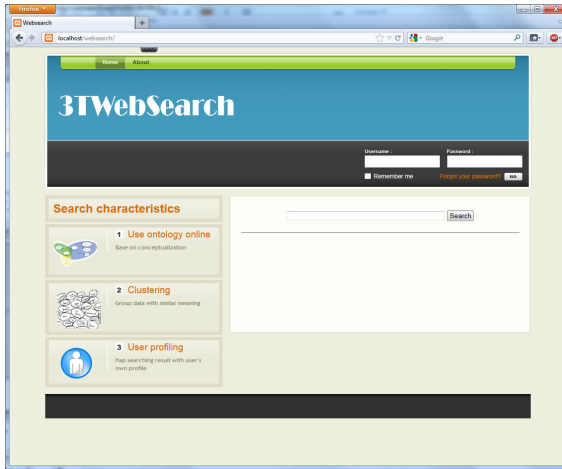


Fig. 3. GUI of user identification

4.2 Step 2 - Ontology Searching and Clustering

In this step, user enters keywords. They are then mapped with ontology online. The results are mixed up and clustering of hits is used and listed into specific group. These processes are shown in Figure 4.

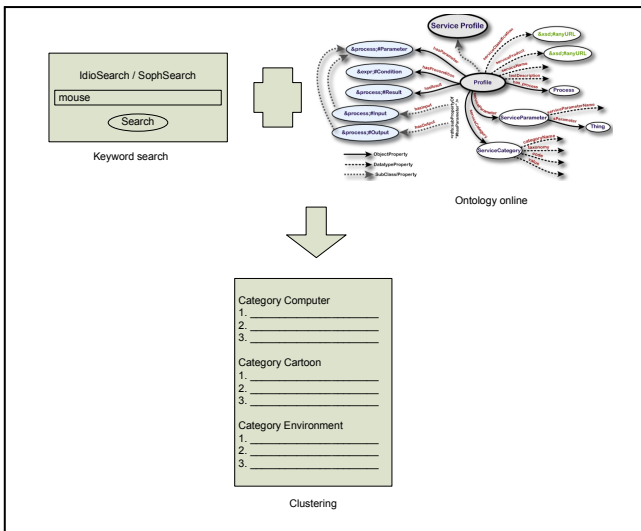


Fig. 4. Web ontology searching and clustering

By using framework in [11], the expected output is shown in Table 4. Categories are listed randomly without considering users' profile. The datasets indicate all users are obtaining the same results. Enhancement using user profiling technique is discussed between Step 3 to 5.

Table 4. Results using Trillo et al.[11] framework

Users	UTHM Staff	Web Category
Yusliza	Yes	Computer
		Cartoon
		Environment
Azma	Yes	Computer
		Cartoon
		Environment
Ziela	No	Computer
		Cartoon
		Environment

4.3 Step 3 - User Profiling Using RDB to RDF Mapping

This process uses Direct Mapping technique. Staff ID entered in Step 1 is used here. It then mapped to UTHM relational database from Table 1 and 2. Structures from these tables are shown in Figure 5 and Figure 6. Mapping process coding which use RDF and SPARQL, query language for RDF is listed in Figure 7.

facID	VARCHAR2 (3)	PRIMARY KEY
facName	VARCHAR2 (50)	

Fig. 5. Faculty table

staffID	VARCHAR2 (10)	PRIMARY KEY
staffName	VARCHAR2 (50)	
facID	VARCHAR2 (3)	FOREIGN KEY

Fig. 6. Staff table

```

<?xml version="1.0" ?>
<rd2rdf xmlns:xyz="http://xyz.com"
        xmlns:rdf="http://www.uthm.edu.my/rdf-syntax-ns#">

<ClassMap ClassName="xyz:fac" GraphName="xyz:FacGraph">
  <SQLDefString>
    Select '<xyz.com/fac/' || facID || '>' AS facURI
          , facNo
          , facName
    from fac
  </SQLDefString>
  <PropertyMap PropertyName="instURI" ColPosInSQLdefString="1" />
  <PropertyMap PropertyName="fac:facID" ColPosInSQLdefString="2" />
  <PropertyMap PropertyName="fac:facName" ColPosInSQLdefString="3" />
  <KeyPropertyMap KeyPropertyName="fac:c_prm_facID" KeyType="Primary">
    <KeyPropertyDef>
      <PropertyName name="fac:facID" posInKey="1" />
    </KeyPropertyDef>
  </KeyPropertyMap>
</ClassMap>

<ClassMap ClassName="xyz:staff">
  <SQLDefString>
    Select '<xyz.com/staff/' || staffID || '>' AS staffURI
          , staffID
          , staffName
          , facID
    from staff
  </SQLDefString>
  <PropertyMap PropertyName="instURI" ColPosInSQLdefString="1" />
  <PropertyMap PropertyName="staff:staffID" ColPosInSQLdefString="2" />
  <PropertyMap PropertyName="staff:staffName" ColPosInSQLdefString="3"
/>
  <KeyPropertyMap KeyPropertyName="staff:c_prm_staffID"
KeyType="Primary">
    <KeyPropertyDef>
      <PropertyName name="staff:staffID" posInKey="1" />
    </KeyPropertyDef>
  </KeyPropertyMap>
  <KeyPropertyMap KeyPropertyName="staff:c_ref_facID"
KeyType="Reference" RefKeyPropertyName="fac:c_prm_facID">
    <KeyPropertyDef>
      <PropertyName name="staff:facNum" posInKey="1" />
    </KeyPropertyDef>
  </KeyPropertyMap>
</ClassMap>
</rd2rdf>

```

Fig. 7. RDF and SPARQL coding to map UTHM database

4.4 Step 4 - RDF to UTHM Ontology Comparison

UTHM ontology as shown in Figure 8 is developed to ensure changes are not done to the database. Modification to the databases will affect current systems since we use actual UTHM datasets. After clustering, the user's faculty captured and mapped in Step 3 is compared with UTHM ontology and find dedicated user's faculty. Field derived from this process is compared with Category in Step 2.

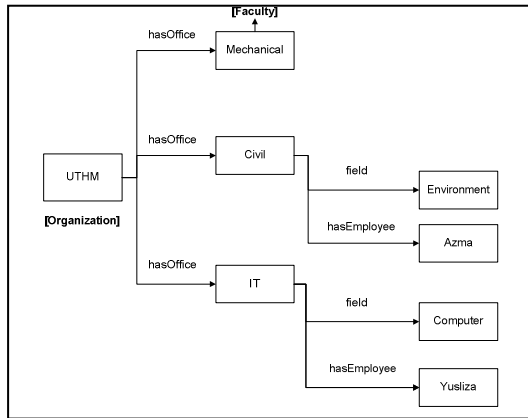


Fig. 8. UTHM Ontology

4.5 Step 5 - Ranking

In this final stage, clustered/categorized hits are ranked depending on user’s data. As shown in Table 5, this Semantic Web Search use entered ID as identification. Name is captured from the RDB. If the user is UTHM staff, the web will get faculty field obtained from Step 4. Clustered activity conducted in Step 3 which produce web categories are compared with results from Step 4. Similar result will give highest rank. Non-similar result but still in the UTHM ontology will be on the lower rank and lastly, non-similar and not in the ontology will be on the lowest level. If the user is not UTHM staff, category will be ranked randomly. Figure 9 shows the expected result in GUI.

Table 5. Web category ranking

ID	Name	UTHM Staff	Field	Web Category	Web Category = UTHM Field	Rank
718	Yusliza	Yes	Computer	Environment	1	2
				Cartoon	0	3
				Computer	1	1
615	Azma	Yes	Environment	Environment	1	1
				Cartoon	0	3
				Computer	1	2
-	Ziela	No	-	Environment	0	1
				Cartoon	0	2
				Computer	0	3

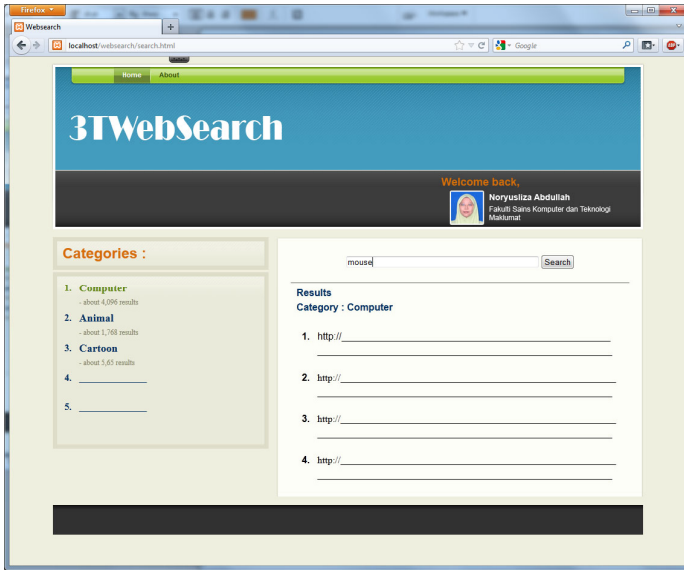


Fig. 9. Semantic Web Search Engine

This framework is based on the previous researchers, Trillo *et al.* [11]. In contrast with our research, only list of categories is given from the online ontologies and clustering processes. Nevertheless, they are mixed up and listed randomly. Excessive numbers of categories will cause confusion. Conversely, we are expected to produce results that are reliable towards user preferences by adding user profiling technique. This technique generates results in Table 5. It produce ranking that does not exist in Table 4.

5 Conclusion

The propose framework of knowledge retrieval using hybrid Semantic Web Search has been discussed. They are three criteria namely online ontology, clustering and user profiling have been used in this research. Enhancement using user profiling criteria is embedded to the current practice which only uses ontology online and clustering. It will give more reliable search results by considering users' own data in RDB. This paper provides the framework, algorithm, datasets structure and the expected result. To produce better illustration, example is enclosed in this paper with detail explanation. This hybrid Semantic Web Search Engine implementation is capable to give the desired result in terms of user's profile.

Acknowledgement. This work is supported by Universiti Tun Hussien Onn Malaysia (UTHM) and Faculty of Computer Science and Information Technology, UTHM. The authors would like to thank Information Technology Centre, UTHM for providing statistic and live data.

References

1. Maier, R., Hadrich, T., Peinl, R.: *Enterprise Knowledge Infrastructures*, 2nd edn. Springer, Berlin (2009)
2. Yao, Y., Zeng, Y., Zhong, N., Huang, X.: Knowledge Retrieval (KR). Paper Presented at the 2007 IEEE/WIC/ACM International Conference on Web Intelligence (2007)
3. Tao, X., Li, Y., Nayak, R.: A knowledge retrieval model using ontology mining and user profiling. *Integrated Computer-Aided Engineering* 15(4), 313–329 (2009)
4. Mikroyannidis, A.: Toward a Social Semantic Web. *Computer* 40(11), 113–115 (2007)
5. d'Aquin, M., Noy, N.F.: Where to publish and find ontologies? A survey of ontology libraries. *Web Semantics: Science, Services and Agents on the World Wide Web* 11(0), 96–111 (2011)
6. d'Aquin, M., Lewen, H.: Cupboard – A Place to Expose Your Ontologies to Applications and the Community. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) *ESWC 2009. LNCS*, vol. 5554, pp. 913–918. Springer, Heidelberg (2009)
7. *Ontology Design Patterns.org (ODP)* (2010), http://ontologydesignpatterns.org/wiki/Main_Page
8. Buitelaar, P., Eigner, T., Declerck, T.: *OntoSelect: A Dynamic Ontology Library with Support for Ontology Selection*. Paper Presented at the International Semantic Web Conference (2004)
9. Thomas, E., Pan, J.Z., Sleeman, D.: *ONTOSEARCH2: Searching Ontologies Semantically (electronic version)* (2008), <http://ceur-ws.org/Vol-258/paper26.pdf>
10. *Schema-cache*, <http://schemacache.com/>
11. Trillo, R., Po, L., Ilarri, S., Bergamaschi, S., Mena, E.: Using semantic techniques to access web data. *Information Systems* 36(2), 117–133 (2011)
12. Jie, T., Limin, Y., Duo, Z., Jing, Z.: A Combination Approach to Web User Profiling. *ACM Trans. Knowl. Discov. Data* 5(1), 1–44 (2010)
13. Antoniou, D., Paschou, M., Sourla, E., Tsakalidis, A.: A Semantic Web Personalizing Technique: The Case of Bursts in Web Visits. In: *Proceedings of the 2010 IEEE Fourth International Conference on Semantic Computing (ICSC)*, pp. 530–535 (2010)
14. Yoo, D.: Hybrid query processing for personalized information retrieval on the Semantic Web (2011)
15. Bussler, C.: Is Semantic Web Technology Taking the Wrong Turn. *IEEE Internet Computing* 12(1), 75–79 (2008)
16. Jiang, H.: Information retrieval and the semantic web. *Proceedings of the Chongqing, China*, pp. V3461–V3463. IEEE Computer Society (2010)
17. Edwards, C.: Analysis: Semantic web's hidden meanings. *Engineering and Technology* 5(16), 52–53 (2010)
18. Cardoso, J.: The semantic web vision: Where are we? *IEEE Intelligent Systems* 22(5), 84–88 (2007)
19. Joo, J.: Adoption of Semantic Web from the perspective of technology innovation: A grounded theory approach. *International Journal of Human Computer Studies* 69(3), 139–154 (2011)
20. Janev, V., Vrane, S.: Applicability assessment of Semantic Web technologies. *Information Processing and Management* 47(4), 507–517 (2010)
21. Wecl, K.: Towards an Ontological Representation of Knowledge on The Web. In: Abramowicz, W. (ed.) *Knowledge-based Information Retrieval and Filtering From the Web*. Kluwer Academic Publisher, USA (2003)

22. Fluit, C., Sabou, M., van Harmelen, F.: Ontology-based Information Visualization. In: Geroimenko, V., Chen, C. (eds.) *Visualizing the Semantic Web: XML-based Internet and Information Visualization*. Springer (2003)
23. Hepp, M.: Ontologies: State of the Art, Business Potential, and Grand Challenges. In: Hepp, M., Leenheer, P.D., de Moor, A., Sure, Y. (eds.) *Ontology Management. Semantic Web, Semantic Web Services, and Business Applications*. Springer, New York (2008)
24. Diez-Rodriguez, H., Morales-Luna, G., Olmedo-Aguirre, J.O.: Ontology-based Knowledge Retrieval. Paper presented at the 2008 Seventh Mexican International Conference on Artificial Intelligence (2008)
25. Hao, Y., Zhang, Y.-F.: Research on Knowledge Retrieval by Leveraging Data Mining Techniques. In: *Proceedings of the 2010 International Conference on Future Information Technology and Management Engineering*, pp. 479–484. IEEE (2010)
26. D’Amato, C., Esposito, F., Fanizzi, N., Fazzinga, B., Gottlob, G., Lukasiewicz, T.: Inductive reasoning and semantic web search. In: *Proceedings of the SAC 2010: Proceedings of the 2010 ACM Symposium on Applied Computing*, pp. 1446–1447. Association for Computing Machinery (2010)
27. D’Aquin, M., Motta, E., Sabou, M., Angeletou, S., Gridinoc, L., Lopez, V., Guidi, D.: Toward a new generation of semantic web applications. *IEEE Intelligent Systems* 23(3), 20–28 (2008)
28. Matthias, H., Gerald, R., Harald, C.G.: A comparison of RDB-to-RDF mapping languages. In: *Proceedings of the Proceedings of the 7th International Conference on Semantic Systems*. ACM, Graz (2011)

Granules of Words to Represent Text: An Approach Based on Fuzzy Relations and Spectral Clustering

Patrícia F. Castro¹ and Geraldo B. Xexéo^{1,2}

¹ Departamento de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, Brasil

² Departamento de Ciência da Computação, IM/UFRJ, Rio de Janeiro, Brasil
{patfuiza,xexeo}@cos.ufrj.br

Abstract. The amount of data available in semi-structured or unstructured format grows exponentially. The area of text mining aims at discovering knowledge from data of this type. Most work in this area uses the model known as bag of words to represent the texts. This form of representation, although effective, minimizes the quality of knowledge discovered because it is not able to capture essential characteristics of this type of data such as semantics and context. The paradigm of granular computing has been shown effective in the treatment of complex problems of information processing and can produce significant results in large-scale environments such as the Web. This paper explores the granulation process of words with a view to its application in the subsequent improvement in text representation. We use fuzzy relations and spectral clustering in this process and present some results.

Keywords: granular computing, fuzzy relation, spectral clustering.

1 Introduction

With the rapid development of the Web, the availability of data in a structured or semi-structured (XML documents, emails, blog posts, academic articles, etc.) has grown exponentially. The discovery of useful knowledge from this data source has been shown to be a complex problem.

Text mining aims to extract knowledge from this type of data by application of techniques of data mining, machine learning, natural language processing, information retrieval and knowledge management. Currently, there is a considerable amount of work with this goal both from databases and from the Web. Most of these works use a common representation electronic text known as bag of words. This representation considers texts as vectors of size n , where n represents the total number of words that appear within a particular collection of documents. Thus, if the word k appears in a text, then the representation of the text will contain a certain value at position k of the corresponding vector. If the word does not appear in the text, this value at position k is equal to zero. There are different approaches to building the collection of words. Likewise, there are different ways of calculating the values of the vector, for example, the number of times the word occurs in the text, the relative

frequency or frequency multiplied by the inverse of the overall frequency of the word, the well-known TF-IDF (term frequency inverse document frequency). These vectors are input to the phase of knowledge construction and validation of algorithms for knowledge discovery, or used in the comparison between the documents in the process of information retrieval. This representation, although effective, is not able to capture many essential features of a text, such as semantics and context. Consequently, if such features are not captured, cannot be considered by the processes that manipulate them, minimizing the quality of discovered knowledge.

A paradigm that arises from the treatment of information, known as granular computing, has attracted the attention of many researchers. According to [1], granular computing gathers a set of theories, methodologies, techniques, and tools, that employ granules to solve complex problems. According to [2], the granules permeate any human task. Humans are constantly abstracting and formulating concepts from these granules, processing these concepts and returning the results of such treatment. To give an example, we can make an analogy with the human capability of dealing with images. At no given moment do we consider the pixels individually. All the time we build groupings of these pixels using some semantics capable of conveying notions of texture, colour, etc. Similarly, when analyzing text, the words are not considered individually. Groupings of these words, representing some semantics, convey their contents. Moreover, humans can perceive the real world through many levels of granularity (abstraction) and can easily alternate between these various levels. Consequently, people abstract and consider only that which serves a specific purpose and ignore that which is irrelevant [3,4]. By being able to focus on different levels of granularity, different levels of knowledge can be obtained as well as a deeper understanding of the structure that is inherent to each type of knowledge. Granular reasoning is, therefore, essential for human intelligence and, according to [5], it can have a significant impact on problem-solving methodologies, especially in large-scale environments such as the Web.

The granulation process is based on the decomposition of objects according to some kind of relationship whereby these objects stay together. The process is inherently fuzzy, vague and imprecise. This paper explores this process, through the use of fuzzy relations. Based on this kind of relationship we use a spectral clustering algorithm in the creation of the granules and present some results.

Apart from this introductory section, this paper presents some related work in section 2. Section 3 provides insights into granulation. Sections 4 and 5 give, respectively, a short review of the concepts of fuzzy relations, and the spectral clustering used as the basis for this work. Sections 5 and 6 present the evaluation methodology and results, respectively. Finally, Section 8 provides some conclusions and the direction of future work.

2 Related Works

Alternative techniques for creating document models have aroused the interest of many researchers. Some approaches are based on the same vector model [6] and others suggest alternative ways. [7] proposes the modeling of texts based on the theory of fuzzy sets. A new algorithm for selecting the terms that characterize a

document is proposed from the fuzzy view point. The experiments produce very favorable results when compared to other methods for selecting terms. [8] presents a new paradigm for mining documents that can exploit the semantic features of documents. The representation scheme, based on graphs, is built through successive stages of syntactic and semantic analysis. A distance measure is presented to determine similarities between the contents of the documents. [9] is based on the cognitive aspects of information retrieval in its proposal. The paper presents the principle of polyrepresentation which represents documents and their respective semantics through various aspects within and between documents. [10], proposes a scheme based on Holographic Reduced Representations (HRR) to encode both the semantic structure and syntactic structure of documents. Finally, similar to the proposal presented in this paper, [11] employs granular computing concepts in the treatment of the problem. In this proposal, the documents assume a representation that includes the knowledge. The granules are formed by sets of keywords with frequent co-occurrence. In this context, other techniques have been proposed. Latent Semantic Analysis (LSA) [12] uses principal component analysis to find groups of words that co-occur. Topic models [13] is a statistical model for discovering abstract topics in document collections. Analysis of formal concepts [14] also uses the evaluation of objects and their relationships in order to identify concepts or topics of interest. We present a new method for the analysis of co-occurrence of words. We also use an algorithm that proves very effective for capturing this type of relationship between words to form granules. We believe this is the greatest contribution of this work.

3 Word Granulation

Granulation means forming aggregates of indiscernible objects. The indiscernibility between these objects can be treated by a similarity function. There are two terms used to denote the main types of similarity between words [15] [16]:

- Paradigmatic: two words are paradigmatically similar if they can be replaced in a specific context. For example, in the sentence and in the context of ‘I bought a car’, the word car can be replaced by automobile without any loss in semantics.
- Syntagmatic: two words are similar if they occur significantly in the same context. For example, the words ‘car’ and ‘transit’ are syntagmatically similar, as they typically occur together in certain contexts.

Despite this distinction, it is relatively rare in published works, and some studies [15] refer to the first type as semantic similarity, and to the second [16] as relatedness similarity. The focus of our work is essentially on the second kind. Both types are computed using different methods and are used in a wide variety of applications. Relatedness similarity is generally measured by employing some statistical or

algebraic tool. In this paper we present a fuzzy approach for the analysis of this similarity.

4 Fuzzy Relation

In this section we present a brief review of the theory of fuzzy relations [17] [18] used in the assessment of the similarity between words and the subsequent creation of granules.

Definition 1. A fuzzy relation between two finite sets $X = \{x_1, \dots, x_u\}$ and $Y = \{y_1, \dots, y_v\}$ is formally defined as a fuzzy binary relation $f: X \times Y \rightarrow [0,1]$, where u and v represent the number of elements in X and Y , respectively.

Definition 2. Given a set of index terms, $T = \{t_1, \dots, t_i\}$ and a set of documents, $D = \{d_1, \dots, d_j\}$, each t_i is represented by a fuzzy set $h(t_i)$ of documents; $h(t_i) = \{F(t_i, d_j) \mid \forall d_j \in D\}$, where $F(t_i, d_j)$ is the membership degree of t_i in d_j .

Definition 3. The fuzzy relationship between words is based on the evaluation of co-occurrence of t_i and t_j in the set D and can be defined as follows:

$$RT(t_i, t_j) = \frac{\sum_k \min(F(t_i, d_k), F(t_j, d_k))}{\sum_k \max(F(t_i, d_k), F(t_j, d_k))} \tag{1}$$

A simplification of the fuzzy RT relation based on co-occurrence of words is given as follows:

$$r_{i,j} = \frac{n_{i,j}}{n_i + n_j - n_{i,j}} \tag{2}$$

where

- $r_{i,j}$ represents the fuzzy RT relation between words i and j
- $n_{i,j}$ is the number of documents containing both the i^{th} and j^{th} words
- n_i is the number of documents containing the i^{th} word
- n_j is the number of documents containing the j^{th} word

5 Spectral Clustering

The spectral clustering technique is characterized by exploring the similarity between all pairs of objects. This technique has proven to be much more effective than more traditional techniques such as the k-means method, for example, which considers only the similarity of the objects to the central elements of their groups [19].

Given n data points $x_1 \dots x_n$, the spectral clustering algorithm constructs a similarity matrix $S \in \mathbb{R}^{n \times n}$, where $S_{i,j} \geq 0$ reflects the relationship between x_i and x_j . It then uses similarity information to group $x_1 \dots x_n$ into k clusters. There are several variants of

spectral clustering. Here we consider the commonly used normalized spectral clustering [20]. An example similarity function is the Gaussian:

$$S_{ij} = \exp \left(-\frac{|x_i - x_j|^2}{2\sigma^2} \right), \tag{3}$$

Where σ is a scaling parameter to control how rapidly the similarity S_{ij} reduces with the distance between x_i and x_j . Consider the normalized Laplacian matrix:

$$L = I - D^{-1/2}SD^{-1/2}, \tag{4}$$

Where D is a diagonal matrix with

$$D_{ij} = \sum_{j=1}^n S_{ij} \tag{5}$$

Note that $D^{-1/2}$ indicates the inverse square root of D . It can be easily shown that for any S with $S_{ij} \geq 0$, the Laplacian matrix is symmetric positive semi-definite. In the ideal case, where the data in one cluster are not related to the data in others, nonzero elements of S (and hence L) only occur in a block diagonal form:

$$L = \begin{bmatrix} L_1 & & \\ & \ddots & \\ & & L_k \end{bmatrix} \tag{6}$$

It is known that L has k zero-eigenvalues, which are also the k smallest ones. These corresponding eigenvectors, written as an $R^{n \times k}$ matrix, are $V = [v_1, v_2, \dots, v_k] = D^{1/2} E$, where $v_i \in R^n, i = 1, \dots, k$ and

$$E = \begin{bmatrix} e_1 & & \\ & \ddots & \\ & & e_k \end{bmatrix}, \tag{7}$$

where $e_i, i=1, \dots, k$ (in different length) are vectors of every one. As $D^{1/2} E$ has the same structure as E , simple clustering algorithms such as k -means can easily cluster V into k groups. Thus, what one needs to do is to find the first k eigenvectors of L (i.e., eigenvectors corresponding to the k smallest eigenvalues). However, in practice the eigenvectors are in the form of $V = D^{1/2}EQ$.

Where Q is an orthonormal matrix. is suggested normalizing V so that

$$U_{ij} = \frac{V_{ij}}{\sqrt{\sum_{r=1}^k V_{ir}^2}}, i = 1, \dots, n; j=1, \dots, k. \tag{8}$$

Each row of U has a unit length. Due to the orthogonality of Q , the above equation is equivalent to

$$U = EQ = \begin{bmatrix} Q_{1,1:k} \\ \vdots \\ Q_{1,1:k} \\ Q_{2,1:k} \\ \vdots \end{bmatrix}, \quad (9)$$

where $Q_{i,1:k}$ indicates the i^{th} row of Q . Then U 's n rows correspond to k orthogonal points on the unit sphere. The n rows of U can thus be easily clustered by k -means or other simple techniques.

Instead of analyzing properties of the Laplacian matrix, spectral clustering algorithms can also be derived from a graph-cut point of view. That is, we partition the matrix according to the relationship between points. Some representative graph-cut methods are Normalized Cut[21], Min-Max Cut [22], and Ratio Cut [23].

6 Evaluation

We have created two distinct bases of texts. The first base contains 200 articles on computational intelligence selected from Google Scholar. These articles are related to 10 distinct subjects: cognition, fuzzy systems, genetic algorithms, neural networks, data mining, knowledge management, machine learning, pattern recognition, optimization and logic. For the second base, 160 articles on text mining/information retrieval were selected from the same site. In this case, eight subjects were used: clustering, latent semantic analysis, information retrieval, ontology, semantics, fuzzy relations, concept extraction and topic models.

Each of these bases was subjected to a pre-processing step where stopwords and words not classified as nouns were removed through application of a tagger available in <http://dragon.ischool.drexel.edu/>. Next, we analyzed the fuzzy correlation between these words, by applying equation 2, presented in section 4. Words and their correlations were subjected to the spectral clustering algorithm with implementation available over <http://www.mathworks.com/matlabcentral/fileexchange/26354-spectral-clustering-algorithms>. The implementation requires information about the value of k . Initially, we adopted k values of 10 and 8 for the first and second base, respectively. The justification for this lies in the fact that we have chosen 10 subjects and, therefore, based on this choice, we can control the groups generated. In a second evaluation, we reduced these values by half: 5 and 4, in each of the bases, respectively. The aim was to examine the clustering algorithm's ability to make generalizations of the words contained in their groups. The algorithm parameters were kept at their default values. Tables 1, 2, 3 and 4 present the most significant words found in each of the clusters generated in each of the scenarios described above.

Table 1. The 10 clusters/granules of Base 1

GRANULE	SUBJECT	KEYWORDS
1	machine learning	computer, aspect, behavior, intelligence, paradigm, years
2	----	exploration, benchmark, architecture, variation, characteristic, interaction, fact
3	neural network	extension, importance, neuron, goal, stability, property, choice
4	knowledge management	storage, knowledge, capability, management, path, business
5	cognition	representation, theory, life, language, cognition
6	pattern recognition	conclusion, input, classification, region, element, application
7	genetic algorithm	population, fitness, optimum, member, algorithm, convergence, solution
8	----	importance, performance, definition, statistics, measurement
9	data mining	attention, data, concept, generalization, addition, relationship
10	----	extraction, example, relations, variable, analysis, satisfaction

Table 2. The 5 clusters/granules of Base 1

GRANULE	SUBJECT	KEYWORDS
1	genetic algorithm / optimization	exploration, performance, fitness, operator, member, algorithm, convergence, solution, population, optimum, crossover
2	neural networks	extension, input, example, property, regression, analysis, neuron, procedure, realization, synthesis, vector, coefficient, manner, applicability

Table 2. (Continued)

3	data mining / knowledge management	user, technique, topic, storage, knowledge, management, capability, information, methodology, data, business, database
4	cognition / logic	behavior, theory, life, paradigm, language, computer, principle, aspect, manipulation, intelligence
5	cognition	protocol, difference, relations, complexity, analysis, problem, role, system, cognition, method, application

Table 3. The 8 clusters/granules of Base 2

GRANULE	SUBJECT	KEYWORDS
01	semantic	evolution, entity, library, management, language, technology, ontology, domain, description, semantics
02	latent semantic analysis	subspace, combination, detection, decomposition, association, retrieval, matrix, effectiveness, vector, collection
03	clustering	example, prototype, constraint, tendency, algorithm, objective, possibility, principle, data, problem,
04	information retrieval	period, kind, property, relations, decomposition, retrieval, information, expansion, criterion, construction
05	concept extraction	extension, representation, evaluation, concept, strategy, selection, explanation, logic, interpretation, identification, text, baseline
06	ontology	mechanism, classifier, correlation, thesaurus, creation, ontology, context, integration, recognition, source, module.
07	fuzzy relations	membership, co-occurrence, set, binary
08	topic models	probability, language, processing, mixture, model, generator

Table 4. The 4 clusters/granules of Base 2

GRANULE	SUBJECT	KEYWORDS
01	semantic/ ontology	development, evolution, entity, library, management, language, version, technology, ontology, methodology, domain, description, semantics, input, mechanism, classifier, correlation, thesaurus, creation, ontology, context, integration, identification, recognition, source, module.
02	latent semantic analysis/ concept extraction	item, user, basis, subspace, combination, detection, decomposition, association, retrieval, matrix, effectiveness, vector, collection, method, extension, representation, evaluation, concept, strategy, selection, explanation, addition, logic, interpretation, identification, text, baseline
03	clustering/ information retrieval	example, prototype, constraint, tendency, algorithm, objective, possibility, finding, principle, data, problem, difficulty, period, user, minimum, kind, property, relations, decomposition, retrieval, information, expansion, criterion, method, construction
04	topic models	probabilistic, language, processing, mixture, model, generative

Aiming to establish a comparison with a well-known approach, we submit on the same basis, an algorithm for latent semantic analysis (LSA). Tables 5 and 6 represent the degree of similarity between the granules generated with the technique proposed in this work and the concepts (granules) obtained with LSA. The contents of each cell in the table represent the percentage of similarity between the granules and concepts. To facilitate the analysis, we highlighted the cells with the greatest similarity measures.

Table 5. Equivalence between granules and concepts for Base 1

		LSA												
		1	2	3	4	5	6	7	8	9	10	11	12	13
G R A N U L E S	1	0.39	0.53	0.50	0.49	0.45	0.42	0.92	0.42	0.42	0.45	0.51	0.32	0.56
	2	0.96	0.41	0.55	0.43	0.45	0.60	0.42	0.56	0.38	0.44	0.32	0.45	0.34
	3	0.46	0.67	0.43	0.40	0.45	0.56	0.50	0.89	0.42	0.34	0.23	0.56	0.42
	4	0.58	0.67	0.76	0.40	0.40	0.54	0.45	0.78	0.23	0.34	0.56	0.56	0.92
	5	0.34	0.45	0.76	0.23	0.40	0.54	0.45	0.78	0.95	0.34	0.56	0.56	0.23
	6	0.39	0.34	0.50	0.87	0.45	0.42	0.67	0.42	0.42	0.67	0.51	0.68	0.45
	7	0.46	0.24	0.43	0.46	0.45	0.56	0.78	0.45	0.42	0.34	0.25	0.56	0.42
	8	0.78	0.85	0.36	0.32	0.45	0.60	0.42	0.15	0.38	0.44	0.47	0.45	0.39
	9	0.39	0.34	0.36	0.87	0.45	0.42	0.67	0.47	0.68	0.67	0.51	0.68	0.76
	10	0.45	0.48	0.50	0.35	0.47	0.42	0.67	0.42	0.65	0.67	0.90	0.68	0.45

Table 6. Equivalence between granules and concepts for Base 2

		LSA									
		1	2	3	4	5	6	7	8	9	10
G R A N U L E S	1	0.39	0.53	0.96	0.49	0.32	0.42	0.51	0.42	0.42	0.45
	2	0.43	0.41	0.32	0.43	0.45	0.60	0.42	0.96	0.38	0.44
	3	0.46	0.67	0.23	0.88	0.68	0.56	0.50	0.47	0.42	0.34
	4	0.58	0.67	0.56	0.40	0.56	0.57	0.45	0.78	0.23	0.87
	5	0.34	0.45	0.56	0.23	0.56	0.54	0.45	0.78	0.95	0.34
	6	0.39	0.34	0.51	0.33	0.68	0.82	0.67	0.42	0.42	0.67
	7	0.46	0.24	0.25	0.46	0.56	0.56	0.89	0.45	0.42	0.34
	8	0.78	0.92	0.47	0.32	0.45	0.60	0.42	0.15	0.38	0.44

7 Results

Looking through Tables 1 and 3, the proposed technique combines words significant enough to present the topics in each of the test bases. In Base 1, for computational intelligence, 7 topics are easily identified from the words associated with their clusters/granules. In Base 2, on text mining / information retrieval, we achieved better results, because the eight subjects that make up the base are easily identified.

The results presented in Tables 2 and 4 show that the technique performs well against the ability of granule generalization contained in the base text. With respect to Base 1 which was tested, we give special emphasis to the grouping of words that describe the topics of genetic algorithms/optimization and data mining/ knowledge management. Such topics are strongly related. The proposed technique shows consistency since it captures these relationships by grouping the words contained in their respective documents.

LSA identified 13 clusters of words for Base 1 text and 10 clusters for Base 2. Despite the greater number of groups, we can see that in all groups of words created with the technique presented in this work, both bases are defined by an LSA equivalence. Thus, we understand that the techniques are equivalent in terms of effectiveness. Although not measured in terms of processing time for each technique, we observed that the technique proposed here performs better than LSA.

8 Conclusion

The paper explored the granulation process based on fuzzy relations of co-occurrence and spectral clustering. The methodology was presented and some preliminary results were shown. These results demonstrate the real applicability of the proposal.

Our next step will be to explore the ability of this technique in the generalization and specialization of granules. This feature will allow the construction of building ontologies with these granules.

We also intend to study a way to allow overlap between the granules produced. The clustering algorithm used does not allow this overlap and we understand that this feature will produce granules much more significant than those produced with the current method.

We believe the introduction of such features will enable the representation of documents whose handling is closer to the human way of dealing with granules, as described in the introduction.

Acknowledgements. The authors would like to thank the financial support of CNPq, CAPES, FAPERJ and Fundação Coppetec.

References

1. Yao, Y.: The Art of Granular Computing. In: Kryszkiewicz, M., Peters, J.F., Rybiński, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 101–112. Springer, Heidelberg (2007)

2. Predycz, W.: *Knowledge-Based Clustering: From Data to Information Granules*. John Wiley & Sons, Hoboken (2005)
3. Yao, Y., Zhong, Y.: *Granular Computing using Information Tables*. In: Lin, T.Y., Yao, Y., Zadeh, L.A. (eds.) *Data Mining, Rough Sets and Granular Computing*, pp. 102–124. Physica, Heidelberg (2002)
4. Yao, Y. *A Ten-year Review of Granular Computing*. In: *Proceedings of IEEE International Conference on Granular Computing*, pp. 734–739 (2007)
5. Zhong, N., et al.: *Towards Granular Reasoning on the Web*. In: *Proceedings of the 2008 Workshop on New Forms of Reasoning for Semantic Web: Scalable, Tolerant and Dynamic (NEFORD 2008), the 3rd Asian Semantic Web Conference, ASWC 2008* (2008)
6. Liu, G.: *The Semantic Vector Space Model (SVSM): A Text Representation and Searching Technique System Sciences*. In: *Proceedings of the Twenty-Seventh Hawaii International Conference on Information Systems: Collaboration Technology Organizational Systems and Technology*, vol. IV, pp. 928–937 (1994)
7. Doan, S., Ha, S., Horiguchi, S.: *A Fuzzy-Based Approach for text Representation in Text Categorization*. In: *14th IEEE International Conference on Fuzzy Systems*, pp. 1008–1013 (2005) ISBN: 0-7803-9159-4
8. Khalid, S.: *A Semantic Graph Model for Text Representation and Matching in Document Mining*. PhD Thesis. University of Waterloo, Canadá (2006)
9. Ingersen, P., Skov, B., Larsen, B.: *Inter and Intra-document Context Applied in Polyrepresentation for Best Match IR*. *Information Processing and Management: an International Journal* 44, 1673–1683 (2008)
10. Fishbein, J.: *Integrating Structure and Meaning Using Holographic Reduced Representation to Improve Automatic Text Classification*. Master Thesis, University of Waterloo (2008)
11. Lin, T.Y.: *Granular Computing and Modeling the Human Thoughts in Web Documents*. In: Melin, P., Castillo, O., Aguilar, L.T., Kacprzyk, J., Pedrycz, W. (eds.) *IFSA 2007. LNCS (LNAI)*, vol. 4529, pp. 263–270. Springer, Heidelberg (2007) ISBN: 978-3-540-72917-4
12. Dumais, S., Landauer, T.: *A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge*. *Psychological Review* 104(2), 211–240 (1997)
13. Steyvers, M., Griffiths, T.: *Probabilistic Topic Models*. In: Landauer, T., et al. (eds.) *Latent Semantic Analysis: A Road to Meaning*. Laurence Erlbaum (2007)
14. Ganter, B., Stumme, G., Wille, R. (eds.): *Formal Concept Analysis*. LNCS (LNAI), vol. 3626. Springer, Heidelberg (2005) ISBN 3-540-27891-5
15. Kozima, T.: *Similarity Between Words Computed by Spreading Activation on an English Dictionary*. In: *Proceedings of the 6th Conference of the European Chapter of the ACL*, pp. 232–239 (1993)
16. Rapp, R.: *The Computation of Word Associations: Comparing Syntagmatic and Paradigmatic Approaches*. In: *Proceedings of COLING 2002* (2002)
17. Chakrabarti, S.: *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann (2003)
18. Haruechaiyasak, C., Shyu, M., Chen, M.L.: *Web Classification Based on Fuzzy Association*. In: *Proceedings of the 25th Annual International Computer Software and Applications Conference (COMPSAC 2002)* (2002)
19. Ng, A., Jordan, M.: *On Spectral Clustering: Analysis and an Algorithm*. In: *Advances in Neural Information Processing Systems*, vol. 14 (2001)

20. von Luxburg, U.: A tutorial on Spectral Clustering. Technical Report 149: Max Planck Institute for Biological Cybernetics (2006)
21. Shi, J., Malik, J.: Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(8), 888–905 (2000)
22. Ding, C., et al.: A Min-max Cut Algorithm for Graph Partitioning and Data Clustering. In: *Proceedings of the First IEEE International Conference on Data Mining (ICDM)*, pp. 107–114. IEEE Computer Society, Washington, DC (2001)
23. Hagen, L., Kahng, A.: New Spectral Methods for Ratio Cut Partitioning and Clustering. *IEEE Transactions Computer-Aided Design* 11(9), 1074–1085 (1992)

Multivariate Time Series Classification by Combining Trend-Based and Value-Based Approximations

Bilal Esmael¹, Arghad Arnaout², Rudolf K. Fruhwirth², and Gerhard Thonhauser¹

¹ University of Leoben
8700 Leoben, Austria
Bilal@stud.unileoben.ac.at,
Gerhard.Thonhauser@unileoben.ac.at
² TDE GmbH
8700 Leoben, Austria
{Arghad.Arnaout,Rudolf.Fruhwirth}@tde.at

Abstract. Multivariate time series data often have a very high dimensionality. Classifying such high dimensional data poses a challenge because a vast number of features can be extracted. Furthermore, the meaning of the normally intuitive term "similar to" needs to be precisely defined. Representing the time series data effectively is an essential task for decision-making activities such as prediction, clustering and classification. In this paper we propose a feature-based classification approach to classify real-world multivariate time series generated by drilling rig sensors in the oil and gas industry. Our approach encompasses two main phases: representation and classification.

For the representation phase, we propose a novel representation of time series which combines trend-based and value-based approximations (we abbreviate it as TVA). It produces a compact representation of the time series which consists of symbolic strings that represent the trends and the values of each variable in the series. The TVA representation improves both the accuracy and the running time of the classification process by extracting a set of informative features suitable for common classifiers.

For the classification phase, we propose a memory-based classifier which takes into account the antecedent results of the classification process. The inputs of the proposed classifier are the TVA features computed from the current segment, as well as the predicted class of the previous segment.

Our experimental results on real-world multivariate time series show that our approach enables highly accurate and fast classification of multivariate time series.

Keywords: Time Series Classification, Time Series Representation, Symbolic Aggregate Approximation, Event Detection.

1 Introduction

Multivariate time series data are ubiquitous and broadly available in many fields including finance, medicine, oil and gas industry and other business domains. The

problem of time series classification has been the subject of active research for decades [1, 7].

The general time series can be defined as follow: A time series T is a series of ordered observations made sequentially through time. We denote the observations by:

$$x_i(t); [i = 1, \dots, n; t = 1, \dots, m]$$

- i is the index of the different measurements made at each time point t ,
- n is the number of variables being observed, and
- m is the number of observations made.

If the time series has only one variable ($n = 1$) then this time series is referred to as univariate, if it has two variables or more ($n > 1$) then it is referred to as multivariate.

One example of multivariate time series is drilling rig data; where many mechanical parameters such as torque, hook load and block position, are continuously measured by rig sensors and stored in real time in the databases. Fig. 1 shows drilling multivariate time series consisting of eight variables.

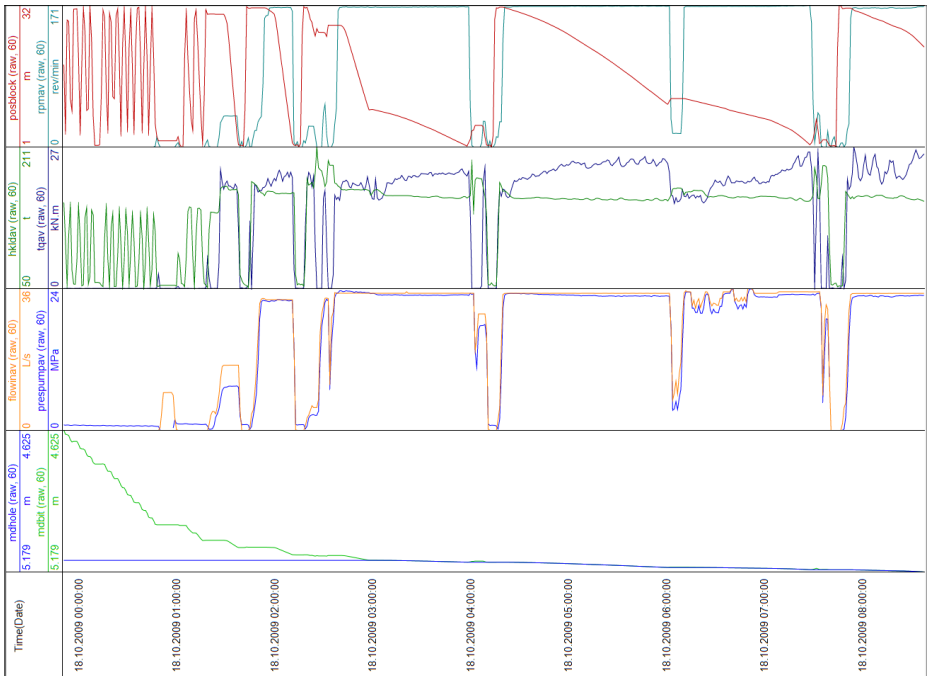


Fig. 1. A multivariate time series of drilling data. This time series consists of eight variables representing eight mechanical parameters measured at the rig.

Multivariate time series classification is a supervised learning problem aimed for labeling multivariate series of variable length. Time series classification can be divided into two types. In the first type (simple classification) each time series is

classified into only one class label, whereas in the second type (strong classification) each time series is classified into a sequence of classes.

This work focuses on the second type of classification. Our approach aims to classify multivariate time series (like the one shown in Fig. 1) into a sequence of operations or classes $op_1(st_1, et_1), \dots, op_n(st_n, et_n)$ where st_i and et_i represent the start time and end time of the operations respectively. Fig. 2 shows the result of such a classification process.

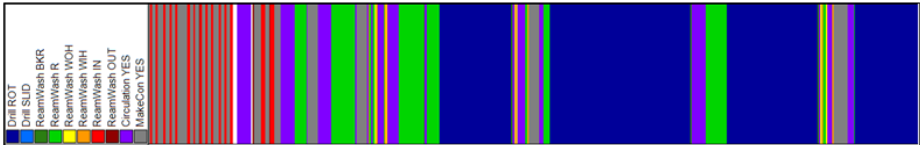


Fig. 2. A sequence of 10 operations with different durations

The main contributions of this work are:

- An approach to represent time series by combining value-based and trend-based approximations (TVA). It extends Symbolic Aggregate Approximation (SAX) [2] by adding new string symbols (U, D and S) to represent the directions of the time series.
- A memory-based classifier for multivariate time series classification. The classifier is trained with the TVA features extracted from our representation. In addition, it uses the previous predicated class as an additional feature to predicate the class of the current segment.

The remainder of the paper is organized as follows: Section 2 introduces the state-of-art techniques for time series representation. Section 3 presents the general framework of our approach. Section 4 explains the details of TVA representation. Section 5 discusses the time series classification. Finally, section 6 presents the experimental results of the proposed approach using real-world data from the drilling industry, and Section 7 concludes the work.

2 State of the Art

Time series datasets are typically very large. The high dimensionality, high feature correlation, and the large amount of noise that can be present in time series, pose a challenge to time series data mining tasks [2]. The high dimensionality of such time series increases both the access time to the data and computation time needed by the data mining algorithms used [8]. Additionally, visualization techniques need to employ data reduction and aggregation techniques to cope with the high volume of data that cannot be plotted in details at once. Furthermore, the very meanings of terms such as “similar to” and “cluster forming” become unclear in high dimensional space [1].

The aforementioned reasons make applying machine learning techniques directly on raw time series data cumbersome. To overcome this problem, the original “raw”

data need to be replaced by a higher-level representation that allows efficient computation on the data, and extracts higher order features [2, 3 and 4].

Several representation techniques, known as dimensionality reduction techniques, have been proposed. This includes the Discrete Fourier Transform (DFT), the Discrete Wavelet Transform (DWT), Piecewise Linear Approximation (PLA), Piecewise Aggregate Approximation (PAA), Adaptive Piecewise Constant Approximation (APCA), Singular Value Decomposition (SVD) and Symbolic Aggregate Approximation (SAX). Choosing the appropriate representation depends on the data at hand and on the problem to be solved. Furthermore, it affects the ease and efficiency of time series data mining [1].

Trend-based and value-based approximations have been used extensively in the last decade. Kontaki et al. [10] propose using PLA to transform the time series to a vector of symbols (U and D) denoting the trend of the series. Keogh and Pazzani [8] suggest a representation that consists of piecewise linear segments to represent a shape; and a weight vector that contains the relative importance of each individual linear segment.

SAX, proposed by Lin et al. [2], is a symbolic approximation of time series. It employs a discretization technique that transforms the numerical values of the time series into a sequence of symbols from a discrete alphabet. The discretization process allows researchers to apply algorithms from text processing and bioinformatics disciplines [2]. SAX has become an important tool in the time series data mining, and has been used for several applications such as time series classification, events detection [5, 6], and anomaly detection [11]. It enables using the Euclidian distance of the discretized subsequences [9], and allows both dimensionality reduction and lower bounding of L_p norms [11].

Although the above mentioned advantages, SAX suffers from some limitations. It does not pay enough attention to the directions of the time subsequences and may produce similar strings for completely different time series. To overcome this problem we propose the TVA representation which extends SAX by adding new string symbols in order to represent the trends of time series.

3 Our Approach

The general framework of the proposed approach is shown in Fig. 3. The given multivariate time series is first divided into a sequence of smaller segments by sliding a window incrementally across the time series. Then, the processing is performed in two phases: representation and classification

- In the representation phase each segment w_i is represented by a pair of characters (v, t) . The first character v represents the linguistic value of the time series and takes one of these values: (a = low), (b = normal), (c = high), etc. The second character t describes the local trend of the time series and takes one of these values: (U = up), (D = down) or (S = straight).
- In the classification phase, a memory-based classifier is trained and used to assign a class label to each segment.

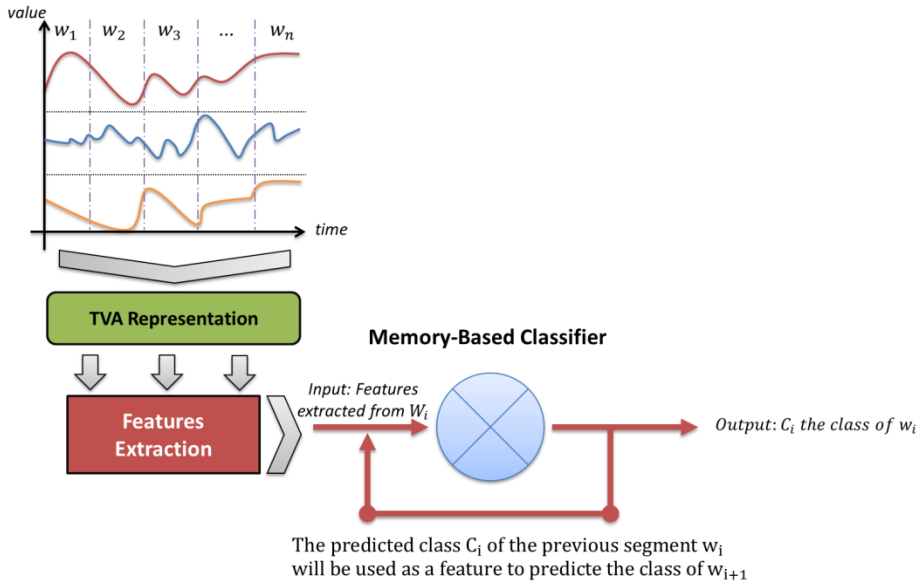


Fig. 3. The general framework of the proposed approach

4 TVA Representation

In the classification phase, we are not interested in the exact numerical values of each data point in the given time series. What we are interested in are the trends, shapes and patterns existing in the data. To recognize these patterns first it is required to discover the simple local trends such as “increase in the hookload” and “decrease in the torque” and to divide the numerical values of the time series into discrete levels such as “high hookload” and “low pressure”.

The TVA representation transforms the numerical values of each variable in the given time series into a sequence of $\langle value, trend \rangle$ pairs. The multivariate time series T is hence transformed as follows:

$$T = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^m \\ x_2^1 & x_2^2 & \dots & x_2^m \\ \vdots & \vdots & \dots & \vdots \\ x_n^1 & x_n^2 & \dots & x_n^m \end{bmatrix} \Rightarrow R = \begin{bmatrix} (v_1^1, t_1^1) & (v_1^2, t_1^2) & \dots & (v_1^s, t_1^s) \\ (v_2^1, t_2^1) & (v_2^2, t_2^2) & \dots & (v_2^s, t_2^s) \\ \vdots & \vdots & \dots & \vdots \\ (v_n^1, t_n^1) & (v_n^2, t_n^2) & \dots & (v_n^s, t_n^s) \end{bmatrix}$$

where R is the matrix that contains the $\langle value, trend \rangle$ pairs, s denotes the number of the segments, v_i^k represents the discrete level of the time series variable i in segment k , and t_i^k represents the trend (direction) of this variable in the segment.

4.1 Value-Based Approximation

In our TVA representation we use the SAX technique to approximate the values of the time series. Two steps should be followed:

- Transforming the given time series T into PAA segments.
- Discretization of the time series based on predefined breakpoints.

Transforming Step

In this step, PAA is used to transform the given time series T of length m into a time series of length s by dividing the original time series into equal-sized segments, and then computing the mean value w_i for each segment i as follows:

$$w_i = \frac{s}{m} \sum_{j=\frac{m}{s}(i-1)+1}^{\frac{m}{s}i} x_j$$

The time series T is represented by a vector of mean values $W = \{w_1, \dots, w_s\}$

Discretization Step

In this step, a further transformation is applied to obtain a discrete representation by producing symbols with equiprobability. The inventors of SAX mentioned that in empirical tests on more than 50 datasets, the normalized subsequences have a highly Gaussian distribution [2]. This enables determining the “breakpoints” that produce equal-sized areas under a Gaussian probability density function. After determining the breakpoints, the time series T is discretized in the following manner: All PAA coefficients that are below the smallest breakpoint are mapped to the symbol “a”, all coefficients greater than or equal to the smallest breakpoint and less than the second smallest breakpoint are mapped to the symbol “b”, and so forth.

Fig. 4 illustrates how the transformation and discretization phases are applied on the data (hook load data). In this example, with $m = 100$ and $s = 10$, the given time series is mapped to the word *hcdacafgfg*.

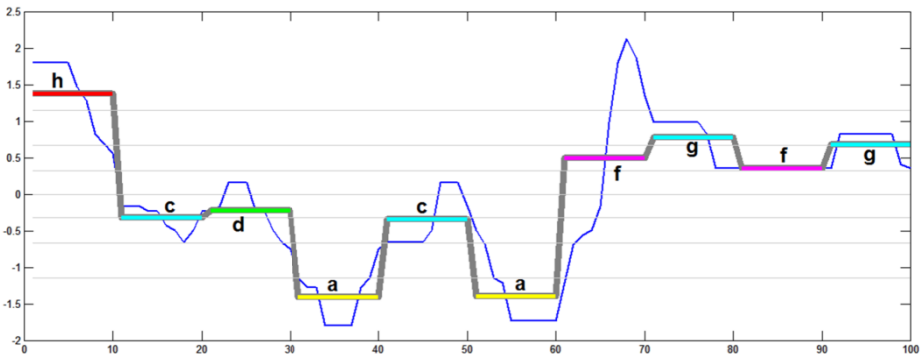


Fig. 4. A time series (blue line) is discretized by first obtaining a PAA approximation (gray line) and then using predetermined breakpoints to map the PAA coefficients into symbols

Indeed, representing the time series, using only the value approximation (SAX), causes a high possibility to miss some important patterns in some time series data. SAX does not pay enough attention to the shapes of the time subsequences and may produce similar strings for completely different time series. Fig. 5 shows an example.

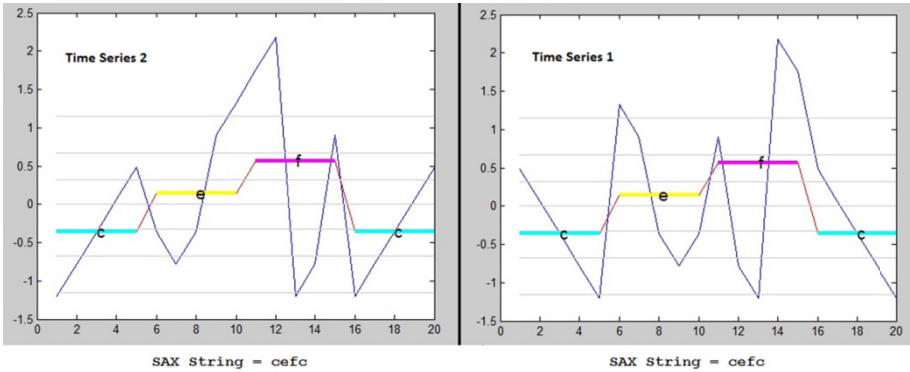


Fig. 5. Two completely different time series that have the same sax string

The above mentioned problem is overcome by adding trend-based approximation beside value-based approximation in order to represent the directions of time series.

4.2 Trend-Based Approximation

We propose using the trends as basis for classifying time series data because these trends form an important characteristic of a time series. In addition, trend-based approximation of time series is closer to human intuition [10].

To generate a trend-based approximation, the least squares method is used to fit a straight line through the set of data points. The least squares method assumes that the best-fit line is the line that has the minimal sum of the squared deviations (least squares error) from a given set of data.

According to the least squares method, the best fitting line has the property that:

$$\sum_{i=1}^n d_i^2 = \sum_{i=1}^n [y_i - f(x_i)]^2 \rightarrow a \text{ minimum}$$

After constructing the lines that fit the data points, the slopes of these lines is calculated, and finally the trend characters U, D or S are computed based on the value of the slope. Fig. 6 illustrates how the above mentioned steps are applied to construct the trend approximation for a part of hook load data.

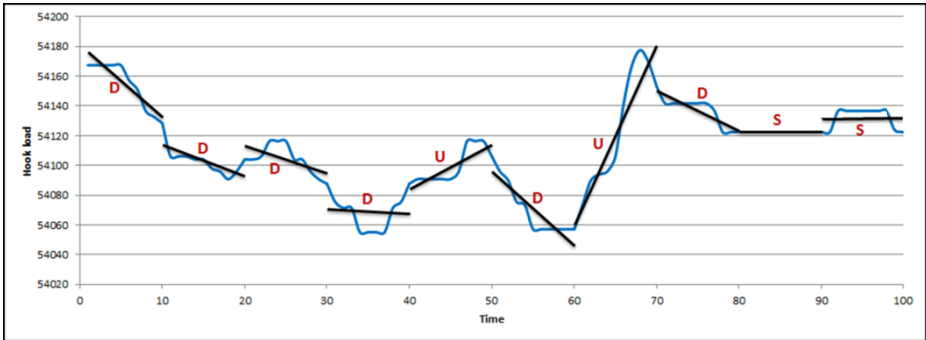


Fig. 6. Trend-based approximation

Using both, the trend-based and the value-based approximation (TVA representation), the given time series is mapped to the string “hDcDdDaDcUaDfUgDfSgS”. Lower case is used to represent SAX values, and upper case is used to represent trends.

5 Time Series Classification

The classification phase starts by extracting a set of features from the TVA representation of the given time series. A feature-vector of fixed length is created for each segment of the time series. The vector has the following form:

$$F^k = (\langle v_1^k, t_1^k \rangle, \langle v_2^k, t_2^k \rangle, \dots, \langle v_n^k, t_n^k \rangle),$$

Where v_i^k and t_i^k represents the SAX character and trend character respectively, and n represents the number of the variables in the given time series. For example, if the given time series has 10 variables and is divided into s segments, then the extracted dataset will have 20 features (columns) and s instances (rows) as shown in Table 1.

After extracting the features, the classifier is trained to assign a class (label) to each segment. In this work we propose a memory-based classifier which takes the previous output of the classification process into consideration. The inputs of the proposed classifier are the feature-vector F^k of the current segment k as well as the predicted class C^{k-1} of the previous segment. Fig. 7 illustrates the pseudo-code of the classification algorithm.

Table 1. An example of the extracted dataset

Seg#	$value_1$	$trend_1$	$value_2$	$trend_2$..	$value_{10}$	$trend_{10}$
1	v_1^1	t_1^1	v_2^1	t_2^1	..	v_{10}^1	t_{10}^1
2	v_1^2	t_1^2	v_2^2	t_2^2	..	v_{10}^2	t_{10}^2
...
s	v_1^s	t_1^s	v_2^s	t_2^s	..	v_{10}^s	t_{10}^s

Classification Algorithm

Input:

- A multivariate time series T of length m

Output:

- A sequence of labels (classes)

Do

- Create an empty sequence of classes SC .
- Divide the original time series T into a set W of smaller equal-sized segments, where $W = \{w_1, w_2, \dots, w_s\}$
- For each segment in W
 - Represent the current segment w_i as mentioned in section 4.
 - Create the feature-vector F_i
 - Get the predicted class of the previous segment c_{i-1}
 - Call the prediction method **predict** (F_i, c_{i-1}) which returns the class c_i of the current segment.
 - Add the predicted class c_i to the sequence SC .
- End For
- For all classes in SC
 - Combine the consecutive equal classes c_1, \dots, c_r in one class C .
 - Set the start time of C equal to the start time of the first class c_1
 - Set the end time of C equal to the end time of the last class c_r
- End For All
- Return SC

End**Fig. 7.** The classification algorithm

Although the memory-based classifiers are simple, as we will show, they improve the classification accuracy significantly. The experimental results show that the average improvement in accuracy is about 8% compared to a traditional classifier.

Many classification techniques can be used to classify the time series. In this work we tested Naïve Bayes, Support Vector Machine, Rule Induction, K-Nearest Neighbor and Decision Trees. Using all these techniques, the classification accuracy was high as we show in the next section. Also, the training time is significantly reduced because the number of extracted features is small.

6 Experimental Results

To evaluate our approach, we tested it with real-world data. Two time series were used in our experiments. Table 2 illustrates these two time series:

Table 2. Time series parameters

Time Series	Length	Frequency	#Variables	#Classes
1	376,840	0.1 Hz	12	10
2	195,808	0.2 Hz	10	9

For all experiments a sequence of ten classes to the first time series, and a sequence of nine classes to the second time series was assigned. Each one of these classes represents one particular operation during drilling. The final output of the classification task is similar to Fig. 2. The proposed approach to represent the data was applied to create the feature space in a first step. Following that, RapidMiner [12] and LIBSVM [13] were used to test the classifiers using the cross validation technique.

Table 3. Classification accuracy

Window Size	Time Series #1		Time Series #2	
	Traditional classifier [%]	Memory-based classifier [%]	Traditional classifier [%]	Memory-based classifier [%]
2	80.40	90.87	92.38	97.88
3	78.23	89.58	93.10	97.71
4	76.62	87.23	92.84	97.51
5	72.87	83.55	93.82	97.67
6	71.40	82.61	93.54	97.43
7	71.03	81.85	93.50	97.14
8	70.25	82.01	92.92	96.70
9	69.86	81.10	92.64	96.35
10	69.41	81.28	92.45	96.33

To measure the improvement that the memory-based classifier provides, two types of classifiers were trained and tested with a varying window size. The first classifier was trained only with TVA features. The second classifier (memory-based classifier) was trained using the extracted features as well as the previously predicted classes as input. Table 3 and Fig. 8 show the results of the Naïve Bayes classifier.

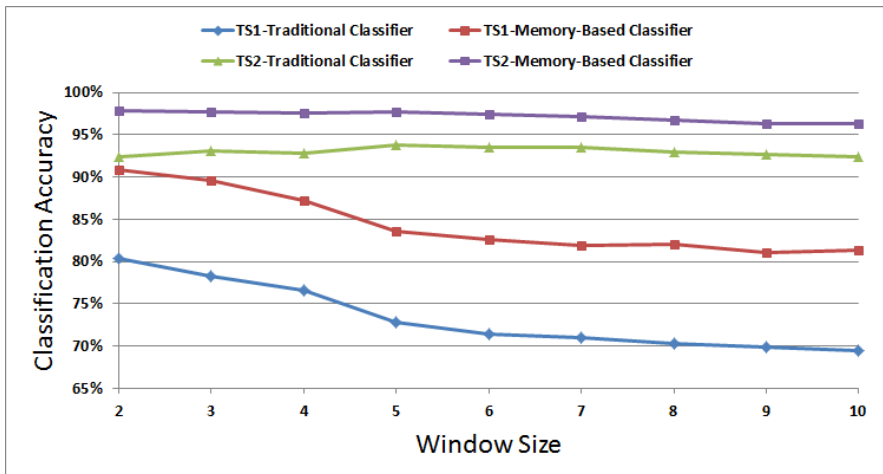


Fig. 8. Classification accuracy vs. window size

Table 4 shows the confusion matrix of one of the experiments in which the Naïve Bayes classifier is applied to time series #2 using a window size of 2.

Table 4. Confusion matrix

	MoveUP	MoveDN	MakeCN	CircHL	ReamDN	DriRot	ReamUP	WashDN	WashUP	Precision [%]
MoveUP	343	8	30	0	0	0	17	0	2	85.7
MoveDN	16	442	4	0	0	0	1	1	1	95.0
MakeCN	12	7	1949	8	0	0	0	20	7	97.3
CircHL	11	1	0	4636	69	27	22	25	61	95.5
ReamDN	1	0	1	13	2155	7	24	30	12	96.0
DriRot	0	0	0	2	5	19028	1	3	1	99.9
ReamUP	5	0	0	5	32	54	2298	1	22	95.0
WashDN	0	6	8	7	19	2	1	920	46	91.1
WashUP	14	1	5	5	2	1	11	27	1632	96.1
Recall [%]	85.3	95.0	97.6	99.1	94.4	99.5	96.7	89.5	91.4	

In addition to Naïve Bayes, four classification techniques were tested. These techniques are: Support Vector Machine (SVM), Rule Induction (RI), Decision Trees (DT) and K-Nearest Neighbor (K-NN). Table 5 summarizes the results.

Table 5. The classification accuracies of different techniques

	SVM	RI	DT	K-NN
Time Series#1	92.9%	91.12%	90.0%	92.8%
Time Series#2	95.23%	98.24%	94.11%	96.9%

7 Conclusion and Future Work

The following conclusion can be drawn from the concepts presented in this paper:

- Representing multivariate time series by combining both the value-based and trend-based approximations leads to reduce the dimensionality of the time series largely.
- The reduced representation can be used as alternative to the time series without losing any important characteristics or patterns exist in the original time series data.
- Memory-based classifiers can improve the classification accuracy of the time series significantly.

Our aim for future work is to improve this approach and use it for writing reports automatically. TVA will be used as an intermediate representation between the numerical values of time series and the human language.

Acknowledgment. We thank TDE Thonhauser Data Engineering GmbH for supporting this work and for the permission to publish this paper.

References

1. Ratanamahatana, C.A., Lin, J., Gunopulos, D., Keogh, E., Vlachos, M., Das, G.: In: Maimon, O., Rokach, L. (eds.) *Data Mining and Knowledge Discovery Handbook 2010*, 2nd edn., pp. 1049–1077. Springer (2010)
2. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In: *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, San Diego, CA, June 13 (2003)
3. Batal, I., Valizadegan, H., Cooper, G.F., Hauskrecht, M.: A Pattern Mining Approach for Classifying Multivariate Temporal Data. In: *IEEE International Conference on Bioinformatics and Biomedicine*, Atlanta, Georgia (November 2011)
4. Batal, I., Sacchi, L., Bellazzi, R., Hauskrecht, M.: Multivariate Time Series Classification with Temporal Abstractions. In: *Proceedings of the Twenty-Second International Florida AI Research Society Conference (FLAIRS 2009)* (May 2009)
5. Onishi, A., Watanabe, C.: Event Detection using Archived Smart House Sensor Data obtained using Symbolic Aggregate Approximation. In: *PDPTA* (2011)
6. Zoumboulakis, M., Roussos, G.: *Escalation: Complex Event Detection in Wireless Sensor Networks*. In: Kortuem, G., Finney, J., Lea, R., Sundramoorthy, V. (eds.) *EuroSSC 2007*. LNCS, vol. 4793, pp. 270–285. Springer, Heidelberg (2007)
7. Wei, L., Keogh, E.: Semi-Supervised Time Series Classification. In: *The Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, SIGKDD (2006)*
8. Keogh, E., Pazzani, M.: An enhanced representation of time series which allows fast and accurate classification clustering and relevance feedback. In: *4th International Conference on Knowledge Discovery and Data Mining*, New York, August 27-31, pp. 239–243 (1998)
9. Hung, N.Q.V., Anh, D.T.: Combining SAX and Piecewise Linear Approximation to Improve Similarity Search on Financial Time Series. In: *Proceedings of the 2007 IEEE International Symposium on Information Technology Convergence (ISITC 2007)*, Jeonju, Korea (2007)
10. Kontaki, M., Papadopoulos, A.N., Manolopoulos, Y.: Continuous Trend-Based Classification of Streaming Time Series. In: Eder, J., Haav, H.-M., Kalja, A., Penjam, J. (eds.) *ADBIS 2005*. LNCS, vol. 3631, pp. 294–308. Springer, Heidelberg (2005)
11. Keogh, E., Lin, J., Fu, A.: HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In: *Proceeding of the 5th IEEE International Conference on Data Mining (ICDM 2005)*, Houston, Texas, November 27-30, pp. 226–233 (2005)
12. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: YALE: Rapid Prototyping for Complex Data Mining Tasks. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2006* (2006)
13. Chih-Chung, C., Chih-Jen, L.: LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2, 27:1–27:27 (2011), Software, <http://www.csie.ntu.edu.tw/~cjlin/libsvm>

Impact of *pay-as-you-go* Cloud Platforms on Software Pricing and Development: A Review and Case Study

Fernando Pires Barbosa and Andrea Schwertner Charão

Programa de Pós-Graduação em Informática,
Universidade Federal de Santa Maria, RS - Brasil
fernando.pires.barbosa@gmail.com, andrea@inf.ufsm.br
<http://www.ufsm.br/ppgi>

Abstract. One of the major highlights of cloud computing concerns the *pay-as-you-go* pricing model, where one pays according to the amount of resources consumed. Some cloud platforms already offer the *pay-as-you-go* model and this creates a new scenario in which the rational computing resource consumption gains in importance. In this paper, we address the impact of this new approach in software pricing and software development. Our hypothesis is that hardware consumption may impact directly on the software vendor profit and thus it can be necessary to adapt some software development practices. In this direction, we discuss the need to revise well-established models such as COCOMO II and some aspects related to requirements engineering and benchmarking tools. We also present a case study pointing that disregarding the rational consumption of resources can generate wastes that may impact on the software vendor profit.

Keywords: Cloud Computing, Software Pricing, Cloud Platform, Software Engineering.

1 Introduction

Cloud computing is a new computing paradigm based on economies of scale, which predicts the existence of a dynamically scalable infrastructure, where resources are allocated and delivered on demand over the Internet [6]. One of cloud computing highlights is its pricing model, also known as *pay-as-you-go*. In this model, IT resources are offered in an unlimited way and one pays an amount according the actual resources used for a certain period (similarly to the energy pricing model) [2].

The cloud vendors offer different kind of services, including *IaaS*, *DaaS*, *PaaS* and *SaaS* [9]. Although one can deploy applications on any of the layers, the more natural option for software developers is the platform as a service (*PaaS*) [2]. Some *PaaS* vendors currently adopt the *pay-as-you-go* model and, in some cases, one can clearly see the financial impact of software optimization techniques applied to resource consumption. In AppEngine platform, for example, studies

indicate that using cache strategy instead of direct database access can reduce by about 20 times the total cost of the operation [2]. Issues like that create a new scenario for software developers, who may have to adapt some of the practices they currently use.

In this new scenario, rational resource consumption become a strategy to reduce the amount of hardware used by applications and therefore reduce the amount to be paid to the vendor's platform. In this paper, we review software pricing issues facing this scenario and analyze some of the practices used in software development, indicating whether and how they are affected by this new reality. The paper is organized as follows: Section 2 focus on software pricing while Section 3 analyze software development aspects. The analysis is focused on: i) software development estimates (with COCOMO II), ii) requirements engineering (ISO/IEC 25010) and iii) benchmarking tools (SPEC). Section 4 presents a case study aiming to identify whether a system developed without concern for the rational resource consumption can generate resource wasting that may result in financial loss if the system is distributed through a *pay-as-you-go* cloud platform. Section 5 presents our final remarks.

2 Software Pricing

Software pricing has been discussed for several years in many ways. Since the '90s there has been studies on establishing a fair price for software [5] and new issues have been addressed recently. One of the recent research topics concerns the differences between the traditional, perpetual license model and the new software as a service (SaaS) model [12]. Beyond the SaaS pricing model issues, different studies on software pricing have been developed, including: models using the value added to client's business [12], studies based on stock market [16], pricing based on cost accounting [19] and use of price sensitivity in order to identify features that should be prioritized [10]. Even with so many different studies, software pricing involves basic elements that can be applied to all models [5], as summarized in Table 1.

Considering the aspects in Table 1, the first item affected by *pay-as-you-go* cloud platforms is number 3. *Revenue Potential*. In the traditional model, the hardware required to run the software is a customer obligation. In the cloud model, it will be probably an obligation applied to software vendor. If the software is based on a *pay-as-you-go* platform, the software vendor company will have part of the revenue gains spent to pay the platform vendor.

Establishing a final software price in this scenario involves estimating not just the software revenue potential, but also the hardware resources required to use it. That changes the way we face the hardware resources throughout the software development process: in the *pay-as-you-go* model, one must design the software to use minimal hardware resources, since it will directly impact your profit. This affects another item in Table 1: number 5. *Estimate software development cost*. Section 3 presents an analysis on software development and the first aspect analyzed, in Section 3.1, is how one of the most used software cost estimation models can face this situation.

Table 1. Software pricing issues [5]

Item	Description	Item	Description
1. Benefit (\$) for customer	Estimating the approximate value that the software will generate for a given customer.	5. Estimates the software development costs	Costs for software development should be estimated and then deducted from the final price in order to find the result.
2. \$Unitary < \$Benefit	Establish a price that is lower than the estimate of the "perceived benefit", so that the customer has a profit perception by purchasing the software.	6. Estimate repairs and maintenance	It should also be provided value to perform repairs and maintenance, especially in software that come with warranty periods.
3. Potential Revenue (\$ Unitary * Potential Sale)	Estimating the potential sale of the software and multiply the unit price for this estimate in order to get the total price of the software.	7. Deduct cost of marketin and distribution	In addition to the costs to develop, should also be estimated expenses to carry out the dissemination, marketing and distribution of software.
4. Potential sales deducted to present value	Sales do not happen all at once, then it is necessary to develop a cash flow of sales in future periods.	8. Losses from piracy	Piracy affects the volume of sales of software and should not be underestimated.

3 Impact on Software Development

Due to changes in software pricing addressed in Section 2, we review some aspects of software development that could also change face to the *pay-as-ou-go* model. In this section, we analyze software cost estimation (with COCOMO II), requirements engineering (with ISO/IEC 25010 and the traditional approach of non-functional requirements based on quality standards) and, at last, benchmarking tools (the traditional SPEC benchmark suites and new research works that are going on).

3.1 Software Cost Estimation (COCOMO)

The first studies on software cost estimation begun in '60s and there has been significant progress since then. Several models have been proposed during the '70s and '80s and some of them have been gradually improved and adapted until today. One of the most used models is COCOMO II (Constructive Cost Model), which is the latest major extension to the original COCOMO (COCOMO 81) model published in 1981 and has several extensions as shown in Fig. 1.

COCOMO II estimates the effort using a person-month value based on 22 items split into 5 software scale drivers and 17 software cost drivers as shown in Table 2. For each item is assigned a value and, the higher this value, the greater the effort required. *Required Software Reliability* (RELY), for example, means the extent to which the software must perform its intended function over a period of time. It ranges from *very low* to *very high*. If the effect of a software failure is only a slight inconvenience, the RELY value is *very low*. If a failure would risk human life then RELY is *very high*.

The *platform* items refers to the target-machine hardware and infrastructure software. *Execution Time Constraint* (TIME) is expressed in terms of the percentage of available execution time to be used by the software. *Main Storage Constraint* (STOR) represents the degree of main storage constraint imposed on the software. Fig. 2 shows the rating ranges for TIME and STOR.

In a *pay-as-you-go* platform, it would be impossible to use the approach based on percentage of available resources as shown in Fig. 2. In a cloud environment,

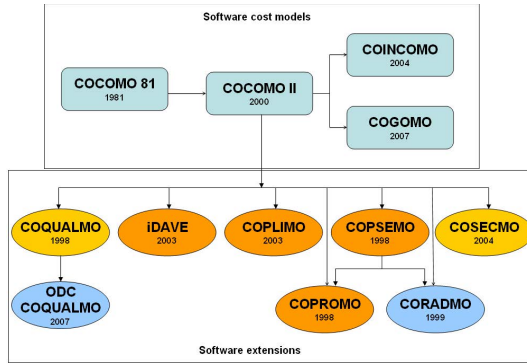


Fig. 1. COCOMO extensions, adapted from [3]. Dates mean the first paper published.

Execution Time Constraint (TIME)

TIME Descriptors:			≤ 50% use of available execution time	70% use of available execution time	85% use of available execution time	95% use of available execution time
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1.00	1.11	1.29	1.63

Main Storage Constraint (STOR)

STOR Descriptors:			≤ 50% use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1.00	1.05	1.17	1.46

Fig. 2. TIME and STORE rating ranges (COCOMO Model Definition Manual [4])

by definition, there is no maximum available resource amount. Resources are provided on demand according to user needs. So it does not make sense to range TIME and STOR based on an available resource percentage.

In a paper on the impact of the cloud model in software engineering [8], the authors propose a change in COCOMO 81, suggesting the creation of a new software class called *Cloud Computing*, to represent the complexity added by cloud platforms. The point is that this steady increase in software complexity is one of the reasons that led replacing COCOMO 81 by COCOMO II. That is, the problem that the article is meant to solve has been solved yet (by COCOMO II). However, even COCOMO II need to be revised. Not just about the cloud environment and its complexity, but mainly about the *pay-as-you-go* model. In such model, the more resources you consume the less its profit. So there is no way to establish a value for the items TIME and STOR based on a percentage of available resources. They must be revised to take into account the rational consumption of resources, even though they are always available. Another question to be addressed on this aspect is: the tools and models currently used allow you know beforehand the amount of resources that will be consumed by a particular

Table 2. COCOMO cost and scale drivers. Highlight on *platform* factor.

	Driver Kind	Driver item	Range					
			Very Low	Low	Nominal	High	Very High	Extra High
Software Cost Drivers (Linear Influence)	Product	Required software reliability	x	x	x	x	x	
		Database size		x	x	x	x	
		Product Complexity	x	x	x	x	x	x
		Developed for Reusability		x	x	x	x	x
		Documentation match to lifecycle needs	x	x	x	x	x	
	Platform	Time Constraint			x	x	x	x
		Storage Constraint			x	x	x	x
		Platform Volatility		x	x	x	x	
	Personnel	Analyst capability	x	x	x	x	x	
		Programmer Capability	x	x	x	x	x	
		Personnel Continuity	x	x	x	x	x	
		Application Experience	x	x	x	x	x	
		Platform Experience	x	x	x	x	x	
		Language and Toolset Experience	x	x	x	x	x	
Project	Use of Software Tools	x	x	x	x	x		
	Multisite Development	x	x	x	x	x	x	
	Required Development Schedule	x	x	x	x	x	x	
Software Scale Drivers (Exponential Influence)	Precedentedness	x	x	x	x	x	x	
	Development Flexibility	x	x	x	x	x	x	
	Architecture / Risk Resolution	x	x	x	x	x	x	
	Team Cohesion	x	x	x	x	x	x	
	Process Maturity	x	x	x	x	x	x	

software? And how do you know if a particular piece of code should or should not be optimized on this perspective? Section 3.2 discusses these issues.

3.2 Requirements and Software Engineering (ISO/IEC 25010)

Requirements engineering is generally accepted to be the most critical and complex process within the software development process [15]. It makes sense, since the requirements are used to describe software features and its behavior. Requirements engineering is mostly performed in the beginning of the software development cycle, but its activities cover the entire cycle in order to detail the software features [15]. Requirements are commonly classified as *functional* and *non-functional*. A functional requirement specifies an action performed by a system without considering its environment. Non-functional requirements describes just the characteristics such as environment, platform, performance, constraints, reliability, etc.

Functional and non-functional requirements are put together in a Software Requirement Specification (SRS) document. It contains details on each feature and information about how the application must interact with the system environment, considering issues such as response time, availability etc. SRS document is the one which will guide the entire software development process.

Being an important issue, requirements are also taken into account by software quality area. ISO/IEC 25010 replaced ISO/IEC 9126 in software quality standards. It defines 8 characteristics for product quality. There are also 31 sub-characteristics as shown in Fig. 3 [11].

ISO 25010 - Software Product Quality							
Functional Suitability	Security	Compatibility	Reliability	Usability	Performance efficiency	Maintainability	Portability
Functional completeness	Confidentiality	Co-existence	Maturity	Appropriateness recognisability	Time behavior	Modularity	Adaptability
Function correctness	Integrity	Interoperability	Availability	Learnability	Resource utilization	Resuability	Installability
Functional appropriateness	Non-repudiation		Fault tolerance	Operability	Capacity	Analysability	Repleaceability
	Accountability		Recoverability	User error protection		Modifiability	
	Authenticity			User interface aesthetics		Testability	
				Accessibility			

Fig. 3. ISO25010 characteristics. Highlight to *efficiency* characteristics.

Performance efficiency characteristic is defined by ISO 25010 as “the *performance relative to the amount of resources used under stated condition*”. That is the quality attribute related to resource consumption. It has 3 sub-characteristics: *time behavior*, related do response time; *resource utilization*, amount and type of resources used; *capacity*, maximum limits of the product (such as concurrent users, size of database, throughput transactions etc.)

Thus, in software engineering, performance and resource consumption requirements are treated as non-functional requirements and have a direct relation with quality attributes, specifically the *Performance Efficiency* characteristic. The *efficiency* measurement is made by objective criteria, which are established in advance and must be attended by the software. An example criteria is: “*on a server with 1GB RAM and two 1,8GHz processors operating with a load of 400 concurrent users, 95% of all requests must return within 5 seconds and 90% of them in up to 3 seconds using up to 50% of CPU and memory available*”.

After detailing the *efficiency* criteria in non-functional requirements at SRS, it is possible to lead measurements to check if software meets the criteria or not. It can be done using forms like the one at Fig. 4. This is how the resource consumption issues are treated in traditional software development process.

Traditional approach induces the software requirement process to identify features with huge concurrent use. That is because one must inform the software designers which features they should provide a special treatment in order to ensure the response time required by quality criteria. In the *pay-as-you-go* model, the amount of resources consumed affects software vendor profit. Therewith, resource consumption is no longer just a matter of quality and must be viewed in a more strategic way. That brings a new issue to software requirements engineering: estimating the amount of resources that will be consumed by the software. Optimize an application to consume the lower amount of resource is different than optimize it to attempt quality criteria such as response time or throughput. The new question to be addressed in the software requirement elicitation process is: “*which features will be used more often?*”.

Internal quality measurement category				
CHARACTERISTIC	SUBCHARACTERISTIC	MEASURESES	REQUIRED LEVEL	ASSESSMEN ACTUAL RESULT
Functionality	Suitability			
	Accuracy			
	Interoperability			
	Security			
	Compliance			
Efficiency	Time behaviour			
	Resource utilisation			
	Compliance			

Fig. 4. Example of form used to software *efficiency* assessment [17]

The answer to these questions indicates which features should be optimized as an strategy to reduce the amount paid to platform vendor. That brings a new concern to software developers about the optimization: *optimizing features with biggest potential to consume resources*. All of that indicates another software development aspect to be reviewed: the performance benchmark tools.

3.3 Benchmarking Tools (SPEC)

Performance evaluation of computer systems has been studied for several years and one of the most appreciated issues are the benchmarking tools. Until the '80s, the main measuring instruments were MIPS and Mflops (both related to CPU speed). But then the systems complexity required new tools [7] and that has led to corporations like SPEC [18]. SPEC was formed to establish, maintain and endorse a standardized set of relevant benchmarks, developing and regulating benchmark suites. SPEC has more than 60 members and also reviews and publishes submitted results from them. Among benchmark suites provided by SPEC, Java benchmarks are the ones with closer relationship to traditional software development. The *SPECJEnterprise2010* benchmark is the most comprehensive of them, since it considers the whole J2EE specification (including Web Server, Application Server, Database Server and JMS System).

SPECJEnterprise2010 is the third generation of the SPEC organization's J2EE industry standard benchmark application. Its performance metric is EjOPS (Enterprise jAppServer Operations Per Second). Earlier generations also used a price/performance metric but it was removed and now one must calculate price/performance separately. That can be done using EjOPS and the BOM (Bill of Materials) used to reproduce the results. Fig. 5 shows how the results are provided by SPEC.

Traditionally, hardware resources are a customer's obligation. In that case, EjOPS metric can help software vendors to suggest hardware specification to be acquired. But if one deploys software through a cloud model, the hardware will probably be in an abstraction layer which is not visible at the customer point of view. Furthermore, the amount spent on these resources will likely to be paid by


 SPECjEnterprise®2010 Result <small>Copyright © 2009-2012 Standard Performance Evaluation Corporation</small>			
Oracle WebLogic Server Standard Edition Release 10.3.6 on Sun Fire X4170 M3		8,310.19 SPECjEnterprise2010 EjOPS	
Submitter: Oracle Corporation		SPEC license # 73	Test date: Feb-2012
Software Products Oracle WebLogic Server Standard Edition Release 10.3.6 Java HotSpot(TM) 64-Bit Server VM on Linux, version 1.7.0_02 Oracle JDBC Driver 11.2.0.3(Thin) Oracle Database 11g Enterprise Edition Release 11.2.0.3	Software Configurations JEE Application Server Emulator Software Config Database Software Config Driver Software Config	Hardware Systems JEE AppServer HW Database Server HW Load Driver & Emulator HW System Configuration Diagram	Benchmark Modifications Configuration Bill of Materials Other Info General Notes Full Disclosure Archive
SUT Configuration			
JEE Server Nodes:	1	DB Server Nodes:	1
JEE Server CPUs:	16 cores, 2 chips	DB Server CPU:	16 cores, 2 chips
JEE Instances:	4	DB Instances:	1

Fig. 5. Example of *SPECJEnterprise2010* result provided by SPEC

the software vendor. In this situation, the EjOPS metric itself makes no sense if there is no related price. The parameters required to evaluate a *pay-as-you-go* cloud platform should be different than ones currently used by SPEC. As shown in Fig. 5, SPEC results include hardware and software specification. That is not relevant in a cloud platform evaluation, which should present something like “how much it will be spent to run a specific workload”. There is also another perspective to be considered: benchmark metrics are generally related to processing power and, in *pay-as-you-go* cloud model, there are other billable items such as total storage used, input and output data transfer. Those items should also be covered by a cloud platform benchmark.

CloudCMP is one of the first efforts in developing a new cloud benchmark suite [13]. The first *CloudCmp* results were published in 2010 at the Conference on Internet Measurement in Australia and the benchmark was publicly released in November, 2011 [14]. *CloudCmp* published data covers some of major cloud vendor such as *Amazon*, *Microsoft*, *Google* and *Rackspace*. Even not addressing only platforms [1], as expected for such research on cloud evaluation, the metrics initially proposed by *CloudCmp* adhere to the new scenario we describe in the present paper. *CloudCmp* compares items like *benchmark finishing time* vs. *cost* or *scaling latency* vs. *cost*, which are closely connected to what would be necessary to choose a cloud platform vendor. But, as the paper itself warns, there is still a lot of work to do. Some of the future work is to build performance prediction models based on *CloudCmp*’s results to enable cloud provider selection for arbitrary apps. That could be used at the beginning of the software development process as a strategy to estimate the amount to be paid to platform vendor.

¹ Only *Google AppEngine* is really a cloud platform (*PaaS*). The other ones are most like *IaaS*.

4 Case Study: SIE ERP System

To study how the changes highlighted in Section 3 could interfere in the outcome of the software development process, we carried out a case study using an ERP system named SIE. This system is targeted to academic institutions and is currently deployed in more than 20 Brazilian universities. SIE development started in mid 1999, using Borland Delphi and a multi-tier architecture where the business rules are processed via RPC calls in one or more centralized application servers [1]. The whole ERP has around 2.500 tables accessed by more than 4.000 applications that work seamlessly to manage different business functions such as academic and student management, contract and inventory management, human resources, finance/accounting, etc.

The specific purpose of our study is to check whether and how a software developed without concern on rational resource consumption can generate wastes that, in a *pay-as-you-go* cloud platform, will directly impact the software vendor profit.

4.1 Case Study Setup

The ideal scenario to perform measurements over a *pay-as-you-go* platform would require that SIE was hosted on a cloud platform. This scenario, however, is not feasible within the scope of this work, because the actual system would need to be developed with the architecture and/or programming language of the platform.

As that scenario is unfeasible, we chose to monitor actual usage of SIE in one institution (Universidade Federal de Santa Maria – UFSM), using its production environment. This approach allows us to collect fairly comprehensive information, since the SIE ERP is widely used in UFSM.

The measurement of effectively consumed resources should track all the billable items of a *pay-as-you-go* platform. Monitoring at this level of detail would not be feasible within the scope of this work, so we chose to monitor the *response time* of each system feature. Then we consider this response time as an indication of resource consumption as follows: the longer response time of a feature, the greater the amount of resources it consumed.

In an ERP like SIE, the response time can be affected by issues such as: CPU usage on the application server; the amount of data transferred between the server and client side; the system’s CPU and disk usage on the database server; and the volume of information stored at database along with the required indexes. Although these aspects may keep some similarity with *pay-as-you-go* billed items, measuring the response time is not a substitute for complete detailed resource monitoring. However, for the purpose of this study, the *response time* can be used without major losses.

4.2 Measurement and Analysis

To perform the measurements and collect data, we modified SIE’s source code to log any RPC call made to the server layer. The log contains, among other

information, the name of the RPC method called and the time each call took to be processed. The change was applied to the UFSM's production environment² and kept alive for a 20 minutes period. This time was enough to collect data on more than 35.000 RPC calls, made by a total of 58 users, who used 62 different system applications, resulting in calls to 602 different RPC methods. With that log data, it was possible to obtain information as shown in Fig. 6, which lists some of the SIE RPC methods and its response time. Taking a look at that list, one observes that the method named *IConsultaLocal.ConsultaAcervoBib* has a considerable total response time (385.336 msec) and it was called 90 times during the monitoring period. Similarly, the method named *ISGCA.GetRotulo* got 6.252 calls and a total response time of 71.225 msec.

NUM_CHAMADAS	TEMPO_TOTAL	TEMPO_MEDIO	TEMPO_MAXIMO	TEMPO_MINIMO	NOME_METODO
90	385336	4.281,51111	116453	204	IConsultaLocal.ConsultaAcervoBib
14	124996	8.928,28571	54328	218	IAutorizacao.GetAplicAutorizadas
13	84824	6.524,92308	13657	952	IDocOcorCurric.AcaoAdaptaCurricAluno
18	79090	4.393,88889	24891	156	ICaixaPostal.GetCxPostalFiltrada
6252	71225	11,39235	14702	0	ISGCA.GetRotulo
418	61101	146,17464	3984	0	IRenovacao.GetRenovacoes
4	59793	14.949,75000	20027	224	IUsuarioGrupo.GetUsuariosNaoGerentes

Fig. 6. Log results obtained for SIE'S RPC

We analyzed the log data aiming to find optimization opportunities that, for some reason, has been disregarded since the beginnings of the software development (more than 10 years ago) and that may impact the total consumption of system resources. The results showed situations as presented in Fig. 7, which lists the 10 RPC methods with higher total response time. The proportions of the 10 methods (in a total of 602 monitored) in relation to the whole system total response time logged reaches 51.39%.

RPC method Name	Calls	Total Response Time	
		Tempo	%
1 IConsultaLocal.ConsultaAcervoBib	90	385.336,0 msec	19,17%
2 IAutorizacao.GetAplicAutorizadas	14	124.996,0 msec	6,22%
3 IDocOcorCurric.AcaoAdaptaCurricAluno	13	84.824,0 msec	4,22%
4 ICaixaPostal.GetCxPostalFiltrada	18	79.090,0 msec	3,93%
5 ISGCA.GetRotulo	6252	71.225,0 msec	3,54%
6 IRenovacao.GetRenovacoes	418	61.101,0 msec	3,04%
7 IUsuarioGrupo.GetUsuariosNaoGerentes	4	59.793,0 msec	2,97%
8 IMulta.CalculaValorAtraso	220	58.058,0 msec	2,89%
9 IItem.ConsisteRenovacao	34	54.400,0 msec	2,71%
10 ISGCA.GetAppConfigFromAplicChamadora	407	54.134,0 msec	2,69%

} 51,39%

Fig. 7. SIE RPC methods with higher total response time (20 min. monitoring)

An analysis of the source code of these method showed situations such as the one of method *IConsultaLocal.ConsultaAcervoBib*, whose optimization would be

² The infrastructure of computers where systems are hosted at UFSM and are accessed by its employees and students.

complex since it has too much possible combinations of parameters and database queries. On the other hand, it also pointed out situations such as the one found on method *ISGCA.GetRotulo*, which could be optimized by implementing a cache system. An analysis of the whole log data pointed out at least another six methods that could be optimized by a cache implementation. Fig. 8 summarizes that.

RPC Method name	Calls	Response Time	Scope	Optimization Opportunity
<i>IConsultaLocal.ConsultaAcervoBib</i>	90	15,33%	Specific Module (Library Management)	Complex: it involves various combinations of parameters and a different logic for each combination
<i>IAutorizacao.GetAplicAutorizadas</i>	14	2,24%	Specific Module (Access Control)	Simple: cache implementation at application server (strong consistency)
<i>ISGCA.GetRotulo</i>	6.252	1,76%	Architecture	Hardworking: review all applications that call it. Possibility of using a local copy of data (low consistence)
<i>ISGCA.GetAppConfigFromAplicChamadora</i>	407	2,75%	Architecture	Simple: cache implementation at application server (strong consistency)
<i>IParInstituicao.GetRecords</i>	702	0,85%		
<i>ITabEstrutura.GetItensTabela</i>	250	0,60%		
<i>ITipoDocPessoa.GetTipoDocPorTipoPessoa</i>	80	2,70%	Specific Module (Main Registering)	
<i>IConfiguracao.GetConfiguracao</i>	689	0,60%	Specific Module (Library)	

Fig. 8. Sample of optimization opportunities found in SIE RPC methods

A previous study suggests that using a cache strategy within the *AppEngine* platform could be up to 20 times cheaper than using direct access to database [2]. Relying on the connection between *response time* and resource consumption, if we just use a simple cache implementation to optimize these seven methods we could reduce their total response time from 346,536 to 17,326 msec. The total response time measured for the whole system during the 20 minutes of monitoring was 2,010,011 msec. So it would represent a reduction of 16.38%. This value of 16.38% represents the savings that the software vendor would have with these optimizations if it was using a *pay-as-you-go* cloud platform.

4.3 Discussion

The relationship between *response time* and resources consumption is not fully accurate and the numbers presented in this section cannot be considered definitive. However, it became more evident that a system developed with traditional methods (like SIE ERP was) and without concern for the rational resource consumption can generate wastings that will impact on software vendor profit if the software is deployed over a *pay-as-you-go* cloud platform. The case study points out that the aspects presented in section 3 are really affected by *pay-as-you-go* cloud platform and it will be necessary to revise some of them. In COCOMO II it would be very useful to revise the cost drivers related to *platform* factor. In benchmarking tools, there is a lot of work ahead to meet the new needs related

to *pay-as-you-go* model. CloudCmp has begun a good work on that but it will be necessary developing more benchmark suites, similarly to the traditional benchmarks from SPEC. ISO/IEC 25010 and requirements engineering, on other hand, may deserve some minor adjustments in the software requirements process to address concern for rational resource usage. All of these changes are summarized in Fig. 9.

Item	Description	At Present	Changes with pay-as-you-go platforms
Pricing	Determining a price for the software	There are several strategies generally divided into two lines: 1- perpetual-use license and 2- rent. In any of these strategies is a set price based on sales potential and revenue-generating software.	The mechanism of perpetual tends to reduce significantly After calculating the potential revenue, this total amount will be reduced by the estimate of resource consumption.
Software Cost Estimation (COCOMO)	Estimates time and cost for software development	Estimates are made that take into account several factors. Of of most used COCOMO. The hardware resources are treated in two main factors: <i>TIME (cpu)</i> and <i>STOR (storage)</i> . Both are measured based on the% of available resources to be used ranging from 50% to 95%.	Measurement based on % of usage resources no longer makes sense on a platform pay-as-you-go, where resources are, by definition, infinite. So, items TIME and STOR of COCOMO must be revised.
Requirements and Software Engineering (ISO/IEC 25010)	Software development process and its connection with resource consumption	Functional and non-functional requirements are eliciting in a decoupled way and then grouped in a SRS document. Non-functional requirements are treated as quality attributes, which beforehand quality criteria (ISO/IEC 25010). In general, there is no concern with the resources consumption during the requirements engineering process. <u>Current Focus:</u> --> What features have concurrent use intensive? --> Optimize those whose response time may be to high	Hardwar resource consumptions get a new strategic importance, beyond the current quality view, since it can impact directly on software vendor profit. It will be necessary identify earlier the resource amount to be consumed by the software. This estimate will likely be made in financial terms. <u>New focus:</u> --> What features will be used more of/en? --> Optimize those whose use tends to consume more resources.
Benchmarking tools	Tools to compare performance of IT solutions	There are benchmarks for many types of software. SPEC is one of organism workin on this area. In general, the benchmarks are focused on measuring the performance of a particular hardware configuration when subjected to execution of a specific type of software. An example is the <i>SPECEnterprise2010</i> , related to enterprise java applications and measured by EJOPS. Metrics like EJOPS can help software developers to guide their clients on aquire the appropriate hardware to host the apps.	Information on the price becomes as or more important than measurements based on "transactions per second". (like EJOPS). There are already some preliminary studies in this area, such as <i>CloudCmp</i> . Pricing information need to be added to the benchmarks (similarly to <i>CloudCmp</i>) and/or new benchmarks need to be built with different measurement items.

Fig. 9. Summary of changes due to *pay-as-you-go* cloud platform

With these changes in mind, we provide in Fig. 10 a schema that can support further studies that aim to adapt software development process to the new scenario. The presented schema is far way from a model proposal. It merely illustrates the idea of identify the expected usage degree of each feature while taking details about that feature and then estimate the amount of resources that will be consumed to decide if it is significant enough to be designed with the maximum optimization possible. This might support the pricing strategy, since the software price must be huge enough to cover all costs of hardware and generate a satisfactory profit margin.

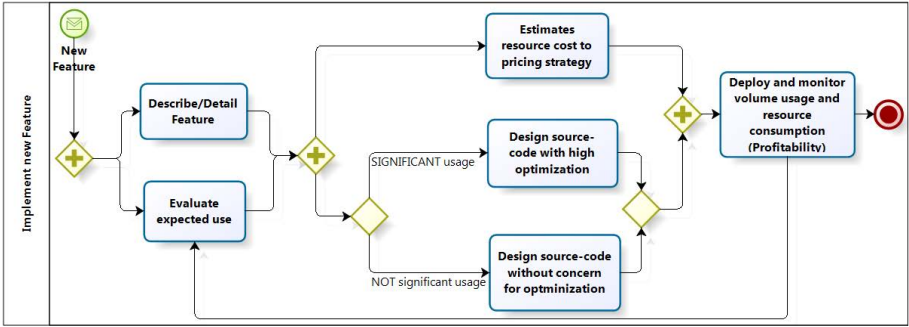


Fig. 10. Draft schema to software requirement development process in a *pay-as-you-go* cloud platform

5 Conclusion

Pay-as-you-go cloud platforms represent a new challenge to software developers: they will need to address the hardware resources in a different way than they are used to. This change is related to the fact that, in these platforms, the hardware consumed by the application usage can directly impact software vendor profit. This reality leads to an approach based on rational resource consumption, which is not the focus of traditional software development.

Not all models and tools currently used in software development are ready to deal with this approach based on rational use of resources. The analysis presented in this paper has pointed out the need to review models such as COCOMO II, as well as processes related to requirements engineering. We also identified the need to review *benchmark* suites and algorithms, such as those provided by SPEC.

The case study showed that software systems developed without concern for the rational resource consumption (as ERP SIE) can lead to resource wastes that will impact on software vendor profit if the software would be hosted on a *pay as-you-go* cloud platform. There is still much work to be done in this area and this paper aims to contribute to raise the problems related to rational resource consumption. Further studies may include reviewing other aspects that deserve attention and were not addressed by this work, such as the use of *pay-as-you-go* platform during the software development itself and some pricing issues related to adding new features to existing software in that scenario.

References

1. Barbosa, F.P.: Projeto e implementação de um framework para desenvolvimento de aplicações em três camadas. Tech. rep., Curso de Ciência da Computação. Universidade Federal de Santa Maria, Santa Maria (2000)
2. Barbosa, F.P., Charão, A.: Uma análise do impacto das plataformas *pay-as-you-go* de computação em nuvem no desenvolvimento e precificação de software. In: Proceedings of the XXXVII Latin American Informatics Conference (XXXVII CLEI) (2011)

3. Boehm, B., Valerdi, R.: Achievements and challenges in Cocomo-based software resource estimation. *IEEE Software* 25(5), 74–83 (2008)
4. Bohem, D.: COCOMO II - model definition manual, version 1.4. Tech. rep., USA (2000), <http://sunset.usc.edu/research/COCOMOII/Docs/modelman.pdf>
5. Dakin, K.: Establishing a fair price for software. *IEEE Software* 12(6), 105–106 (1995)
6. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: *Grid Computing Environments Workshop, GCE 2008*, pp. 1–10 (November 2008)
7. Giladi, R., Ahitav, N.: SPEC as a performance evaluation measure. *Computer* 28(8), 33–42 (1995)
8. Guha, R., Al-Dabass, D.: Impact of web 2.0 and cloud computing platform on software engineering. In: *2010 International Symposium on Electronic System Design (ISED)*, pp. 213–218 (December 2010)
9. Motahari-Nezhad, H.R., Bryan Stephenson, S.S.: Outsourcing business to cloud computing services: Opportunities and challenges. Tech. rep., USA (February 2009), <http://www.hpl.hp.com/techreports/2009/HPL-2009-23.pdf>
10. Harmon, R., Raffo, D., Faulk, S.: Incorporating price sensitivity measurement into the software engineering process. In: *Portland International Conference on Management of Engineering and Technology, PICMET 2003. Technology Management for Reshaping the World*, pp. 316–323 (July 2003)
11. ISO: ISO/IEC 25010. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE). ISO (2011)
12. Kamdar, A., Orsoni, A.: Development of value-based pricing model for software services. In: *11th International Conference on Computer Modelling and Simulation, UKSIM 2009*, pp. 299–304 (March 2009)
13. Li, A., Yang, X., Kandula, S., Zhang, M.: CloudCmp: comparing public cloud providers. In: *Proceedings of the 10th Annual Conference on Internet Measurement, IMC 2010*, pp. 1–14. ACM, New York (2010), <http://doi.acm.org/10.1145/1879141.1879143>
14. Li, A., Yang, X., Kandula, S., Zhang, M.: CloudCmp - pitting cloud against cloud (2011), <http://cloudcmp.net/download>
15. Pandey, D., Suman, U., Ramani, A.: An effective requirement engineering process model for software development and requirements management. In: *International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom)*, pp. 287–291 (October 2010)
16. Qin, W., Ru-xiang, W.: Research of military software pricing based on binomial tree method. In: *3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, vol. 9, pp. 628–632 (July 2010)
17. SEI/PSM: Software quality requirements and evaluation, the ISO 25000 series (2004), <http://www.psmc.com/Downloads/TWGFeb04/04ZubrowISO25000SWQualityMeasurement.pdf>
18. SPEC: Standard performance evaluation corporation (2011), <http://www.spec.org/>
19. Zheng, Y., Cao, R., Sun, W., Zhang, K., Jiang, Z.: Practical application of FDC in software service pricing. In: *IEEE International Conference on e-Business Engineering, ICEBE 2006*, pp. 352–357 (October 2006)

Resilience for Collaborative Applications on Clouds

Fault-Tolerance for Distributed HPC Applications

Toàn Nguyễn and Jean-Antoine Désidéri

Project OPALE

INRIA

38334 Saint-Ismier, France

{Toan.Nguyen, Jean-Antoine.Desideri}@inria.fr

Abstract. Because e-Science applications are data intensive and require long execution runs, it is important that they feature fault-tolerance mechanisms. Cloud and grid computing infrastructures often support system and network fault-tolerance. They repair and prevent communication and software errors. They allow also checkpointing of applications, duplication of jobs and data to prevent catastrophic hardware failures. However, only preliminary work has been done so far on application resilience, i.e., the ability to resume normal execution following application errors and abnormal executions. This paper is an overview of open issues and solutions for such errors detection and management. It also overviews the implementation of a workflow management system to design, deploy, execute, monitor, restart and resume distributed HPC applications on cloud infrastructures in cases of failures.

Keywords: High-Performance Computing, Cloud Computing, Distributed Computing, Scientific Applications, Workflows, Resilience.

1 Introduction

Scientific applications are required today to design, simulate, optimize and manufacture artifacts, ranging from nanotubes to electronic devices and cruisers to airliners.

In order to design quickly these artifacts, long running simulations are executed. For example, multi-discipline scenarios are implemented, where hydraulic, thermic, fluid and electromagnetic simulation software collaborate for the design of nuclear plants.

Long running executions lasting days and even weeks on large HPC clusters suffer from reliability problems concerning the hardware and software infrastructures [15][17]. Fault-tolerance mechanisms are therefore required. They tend to be multi-level, each aspect corresponding to a different level with its specific sources of errors: network communications, distributed middleware, operating systems, collaborating application codes.

The efforts on application errors tend to be the focus of active research today. This is due in part to optimization concerns, and to another part for fault-tolerance concerns [16].

Optimization concerns target the speed-up of CPU and data intensive demanding applications [3]. Parallelization techniques take advantage today of multi-core super-computers. However the 100K+ multi-core HPC clusters today are error-prone and their mean-time between failures is in the order of minutes [13]. Therefore, effective and low-overhead fault-tolerance application algorithms and codes are necessary.

Further, applications misbehavior and errors have multiple origins, which are not necessarily programming errors. They might originate in unforeseen data configurations, especially in simulation applications, unexpected data values, unpredictable behaviors in case of multiple errors cumulating abnormalities, etc.

Important efforts are required to handle these complex abnormal application situations [4][8][14].

This paper addresses the management of application errors and abnormal behavior. It defines terms (section 2), addresses open issues and solutions for error detection (Section 3) and error management (Section 4). It sketches also implementation issues using a workflow management system (Section 5). Section 6 is a conclusion.

A prototype system based on a distributed workflow platform for the design, deployment, execution and monitoring of HPC applications is briefly described. The platform features resilience capabilities to address the application runtime errors.

2 Definitions

Because many terms are used in the fault-tolerance area, we give in this section a definition of various terms used in the domain and pave the way for an ontology of the required concepts.

An interesting definition of errors, faults and failures is given in a system such as Apache's ODE [11], system failures and application faults address different types of errors.

2.1 Errors

The generic term *error* is used to characterize abnormal behavior, originating from hardware, operating systems and applications that do not follow prescribed protocols and algorithms. Errors can be fatal, transient and warnings, depending on their criticality level. Because sophisticated hardware and software stacks are operating on all production systems, there is a need to classify the corresponding concepts (Figure 1).

2.2 Failures

A *failure* to resolve a DNS address is different from a process fault, e.g., a bad expression. Indeed, a system failure does not impact the correct logics of the application process at work, and should not be handled by it, but by the system error-handling

software instead: “failures are non-terminal error conditions that do not affect the normal flow of the process” [11].

2.3 Faults

However, an activity can be programmed to throw a *fault* following a system failure, and the user can choose in such a case to implement a specific application behavior, e.g., a number of activity retries or its termination.

Application and system software usually raise *exceptions* when faults and failures occur. The exception handling software then handles the faults and failures. This is the case for the YAWL workflow management system [19][20], where specific *exlets* can be defined by the users [21]. They are components dedicated to the management of abnormal application or system behavior (Figure 2). The extensive use of these exlets allows the users to modify the behavior of the applications in real-time, without stopping the running processes. Further, the new behavior is stored as a component workflow which incrementally modifies the application specifications. The latter can therefore be modified dynamically to handle changes in the user requirements.

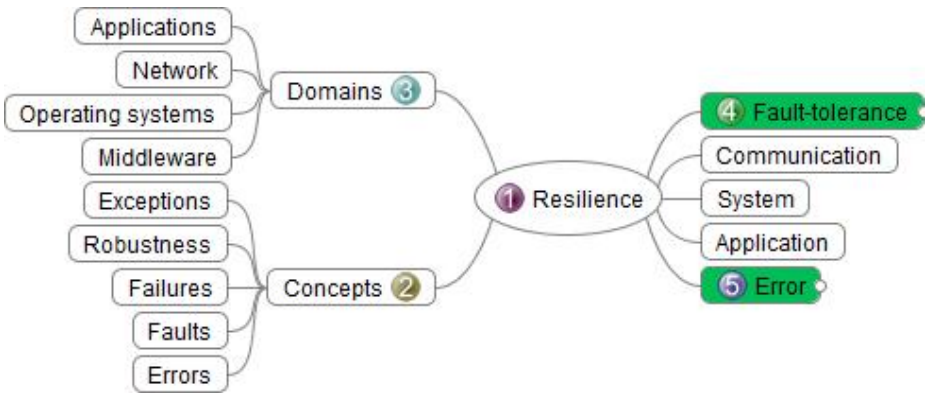


Fig. 1. Resilience domains and concepts

2.4 Fault-Tolerance

Fault-tolerance is a generic term that has long been used to name the ability of systems and applications to handle errors. Transactional systems for example need to be fault-tolerant [9]. Critical business and scientific applications need to be fault-tolerant, i.e., to resume consistently in case of internal or external errors.

2.5 Checkpoints

Therefore *checkpoints* need to be designed at specific intervals to backtrack the applications to consistent points in the application execution, and restart be enabled from there. They form the basis for recovery procedures.

In the following, we call checkpoint for a particular task the set including task definition, parameter specifications and data associated to the task, either input data or output data and the parameter values.

This checkpoint definition does not include the tasks execution states or contexts, e.g., internal loop counters, current array indices, etc. Therefore, we assume that checkpointed tasks are stored stateless. This means that interrupted tasks, whatever the reasons and errors, cannot be restarted from their exact execution state immediately prior to the errors.

2.6 Recovery

We assume therefore that the *recovery* procedures must restart the failed tasks from previously stored elements in the set of existing tasks checkpoints. A consequence is that failed tasks cannot be restarted on the fly, following for example a transient non fatal error. They must be restarted using previously stored checkpoints.

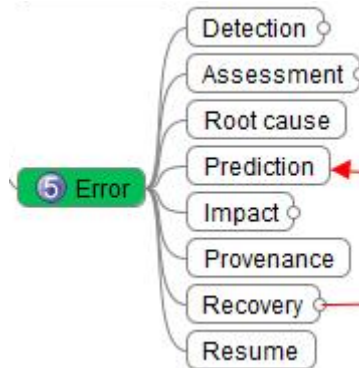


Fig. 2. Error management

2.7 Resilience

Application *resilience* is the property of software that are able to survive consistently from data and code errors (Figure 2). This area is a major concern for complex numeric software that deal with data uncertainties. This is particularly the case for simulation applications [7].

This is also a primary concern for the applications faced to system and hardware errors. In the following, we include both (application external) fault-tolerance and (internal) robustness in the generic term resilience [1].

Therefore we do not follow here the definition given in [17]: “*By definition a failure is the impact of an error itself caused by a fault.*”

But we fully adhere to the following observation: “*the response to a failure or an error depends on the context and the specific sensitivity to faults of the usage scenarios, applications and algorithms*” [17].

3 Error Detection

3.1 Error Characterization

We address in this paper application errors, e.g., out of bounds data values, undefined parameters, execution time-outs, result discrepancies and unexpected values. We do not address communication, hardware and operating systems errors. We suppose that they are handled by the appropriate fault-tolerance sub-systems, which might automatically correct some of them or take appropriate corrective action, e.g., re-routing lost messages. We also suppose that these errors can be signaled to the application-level software by the appropriate raising of exceptions and posting of signals. Thus, the applications can take whatever actions are needed, e.g., re-executing tasks on other resources in case of network partition, out-of-memory execution, etc. This can be defined by the application designers and even by the application users at runtime.

The early characterization of errors is difficult because of the complex software stack involved in the execution of multi-discipline and multi-scale applications on clouds. The consequence is that errors might be detected long after the root cause that initiated them occurred. Also, the error observed might be a complex consequence of the root cause, possibly in a different software layer.

Similarly, the exact tracing and provenance data may be very hard to sort out, because the occurrence of the original fault may be hidden deep inside the software stack.

Without explicit data dependency information and real-time tracing of the components execution, the impacted components and associated results may be unknown. Hence there is a need for explicit dependency information [10].

3.2 Error Ranking

The ranking of errors is dependent on the application logic and semantics (e.g., default values usage). It is also dependent on the logics of each software layer composing the software stack. Some errors might be recoverable (unresolved address, resource unavailable...), some others not (network partition...). In each case, the actions to recover and resume differ: ignore, retry, reassign, suspend, abort...

In all cases, resilience requires the application to include four components:

- a monitoring component for (early) error detection,
- a (effective) decision system, for provenance and impact assessment,
- a (low overhead) checkpointing mechanism,
- an effective recovery mechanism.

Further, some errors might be undetected and transient. Without explicit data dependency information and real-time tracing of the components execution, the impacted components and associated results may be unknown. Hence there is a need for explicit dependency information between the component executing instances and between the corresponding result data [12].

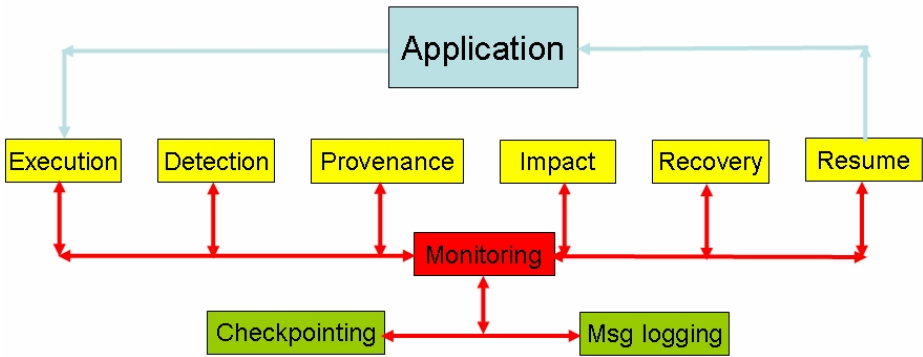


Fig. 3. Resilience sub-system

3.3 Resilience Sub-system

A sub-system dedicated to application resilience includes therefore several components in charge of specific tasks contributing to the management of errors and consistent resuming of the applications (Figure 3). First, it includes an intelligence engine in charge of the application monitoring and of the orchestration of the resilience components [5]. This engine runs as a background process in charge of event listening during the execution of the applications. It is also in charge of triggering the periodic checkpointing mechanism, depending on the policy defined for the applications being monitored [22]. It is also in charge of triggering the message-logging component for safekeeping the messages exchanged between tasks during their execution. This component is however optional, depending on the algorithms implemented, e.g., checkpointing only or hybrid checkpoint-message logging approaches. Both run as background processes and should execute without user intervention. Should an error occur, an error detection component that is constantly listening to the events published by the application tasks and the operating system raises the appropriate exceptions to the monitoring component. The following components are then triggered in such error cases: an optional provenance component which is in charge of root cause characterization, whenever possible. An impact assessment component is then triggered to evaluate the consequences of the error on the application tasks and data, that may be impacted by the error. Next, a recovery component is triggered in charge of restoring the impacted tasks and the associated data, in order to re-synchronize the tasks and data, and restore the application to a previous consistent state. A resuming component is finally triggered to deploy and rerun the appropriate tasks and data on the computing resources, in order to resume the application execution. In contrast with approaches designed for global fault-tolerance systems, e.g., CIFTS [15], this functional architecture describes a sub-system dedicated to application resilience. It can be immersed in, or contribute to, a more global fault-tolerance system that includes also the management of system and communication errors.

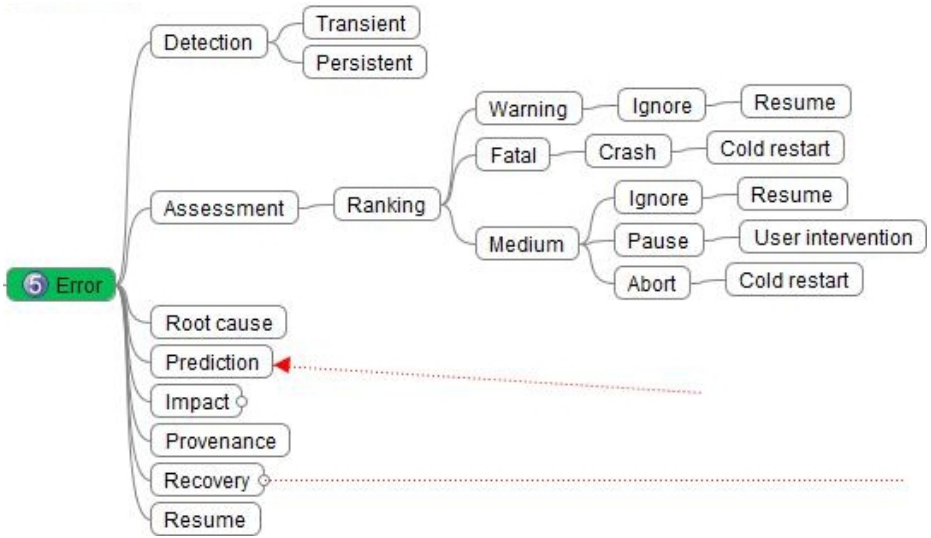


Fig. 4. Error detection and assessment

4 Error Management

Many open issues are still the subject of active research concerning application resilience. The paradigm ranges from code and data duplication and migration, to the monitoring of application behavior, and this includes also quick correctness checks on partial data values, the design of error-aware algorithms, as well as hybrid checkpointing-message logging features (Figure 3). We focus here only on application errors. We do not address hardware, systems and communication errors. We suppose that these errors are fully treated by the appropriate system components [15][22]. We further suppose that they can be signaled to the applications by some exception events. This allows handling the consequences of the errors, e.g., communication failures, by the appropriate application resilience sub-system (Section 3.3).

The baseline is (Figure 4):

- the early detection of errors
- root cause characterization
- characterization of transient vs. persistent errors
- the tracing and provenance of faulty data
- the identification of the impacted components and their associated corrupted results
- the ranking of the errors (warnings, fatal, medium) and associated actions (ignore, restart, backtrack)

- the identification of pending components
- the identification and purge of transient messages
- the secured termination of non-faulty components
- the secure storage of partial and consistent results
- the quick recovery of faulty and impacted components
- the re-synchronization of the components and their associated data
- the properly sequenced restart of the components

Each of these items needs appropriate implementation and algorithms in order to orchestrate the various actions required by the recovery of the faulty application components.

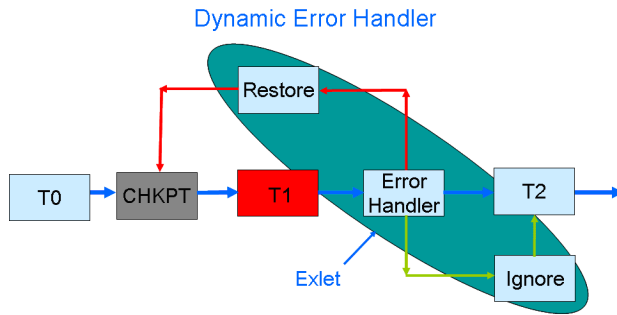


Fig. 5. Error handler

4.1 Detection

Error Detection. Hardware and system fault-tolerance mechanisms can intercept errors [12]. Applications errors however must be explicitly taken into account in the code. This impacts severely the programming efforts and needs important design and re-programming efforts for existing codes [13].

Error Characterization. Similarly, error characterization is heavily dependent on the application logics [14]. It allows for error ranking, ranging from warnings to fatal. This is necessary to fine tune the fault-tolerance and resilience capabilities to the application and user requirements.

Root Cause Detection. Root cause characterization and provenance information is the most difficult part in complex applications and systems. Most of the time, even sophisticated tracing mechanisms will fail to provide an accurate characterization of the multiple root causes that provoke errors and abnormal application behavior [17].

Impact Assessment. The next important step is the assessment of the error impact. This includes the impact on the subsequent tasks, on the data, and the evaluation of error propagations. Further, a domino effect is that the errors can impact the messages exchanged and in transit between tasks as well as the advent of the pending tasks. This is detailed in the following sections (4.2, 4.3).

4.2 Impacted Tasks and Data

Impacted Tasks. The application definition provides a detailed dependency relationship between tasks and data. It should therefore be straightforward to characterize the impacted tasks and data. However, the latency between error occurrence and their actual detection makes it difficult to precisely point out the exact time and location when an error occurred, particularly in distributed systems. Therefore, impacted tasks and data can barely be defined without an undefined uncertainty. This paves the way for drastic backtracking policies and restarts. However, optimized checkpointing schemes, e.g., asymmetric [1], multi-level [18] and encoded checkpoints [22], alleviate somehow crude backtracking and checkpointing approaches by reducing their overhead, in both CPU and storage terms.

Corrupted Data. Similarly, corrupted data can originate from application errors and from error propagation (Figure 6):

Application errors. Computation errors on correct data will produce erroneous results, e.g., specification, algorithmic, programming errors. They can be spotted and corrected with unpredictable delays. Performance and overhead issues are not necessarily fundamental here because CPU and data demanding tasks might have to be backtracked and re-executed, incurring potentially very long delays.

Error propagation. Correct computations performed on previously polluted data may generate random errors on data processed subsequently. Errors cannot in this case be pointed out immediately, if at all. Restarting the application components from ancestor tasks might be a necessary option here. The exact and most accurate restart location may in some cases be difficult to characterize. Policy requirements and implementations are in this case the last resort.

4.3 Impact

Transient Messages. Transient messages are potentially emitted before a component failure. Identifying such data might be very difficult in distributed computing and collaborative codes. Indeed, failed tasks might have sent unknown numbers of messages and data to a potentially unknown number of descendant tasks, depending on the point of failure. Time-outs might here be necessary to consider transient messages to reach their destinations. Purging all these messages is necessary to backtrack to a previous consistent checkpoint.

Pending Tasks. Pending tasks are in contrast easily characterized since they are waiting for incoming data or events raised by ancestor tasks. Pause and resuming of such tasks is an option, without systematically calling for their cold restart from a previous checkpoint. Opportunistic checkpoints might here be interesting to store already

produced data and application state. This is related to asymmetric checkpoints [1], where the users define points of interest in the application runs where checkpoints and snapshots must be stored in order to prevent potential catastrophic failures later. So, CPU, storage and communication demanding tasks will in such cases be saved without the need to restart them later in case of application errors.

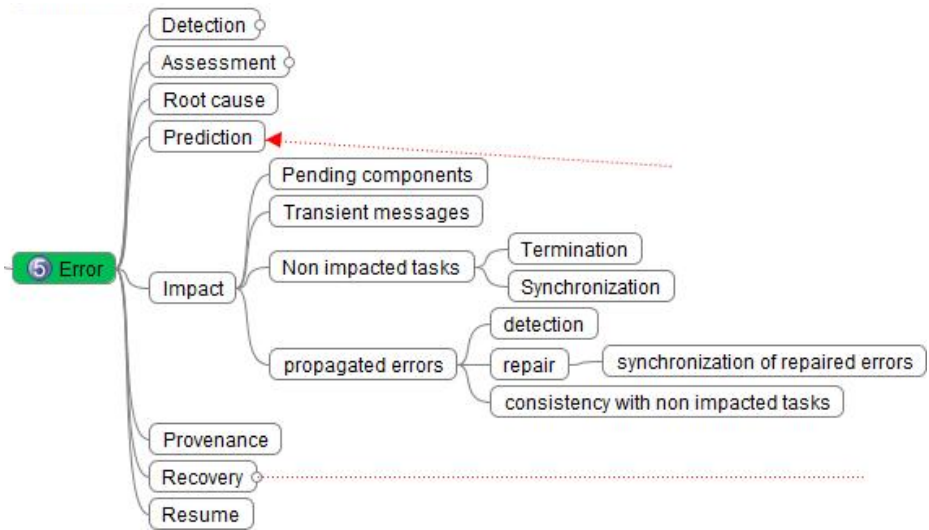


Fig. 6. Error impact

4.4 Recovery

Termination of Non-faulty Tasks. As mentioned above, recovery of non-faulty tasks is straightforward if they are not directly linked to faulty tasks, or if they are explicitly waiting for incoming data or events. If they are directly linked to failed tasks, i.e., processing data produced by failed tasks, restarting them with the failed tasks may be necessary. Indeed, without a sophisticated control of the data exchanged between tasks, it may be impossible to characterize the subsets of data already processed correctly by subsequent tasks. This is also the case when using data pipelining between tasks. In this case, restarting the tasks from the beginning is necessary. Further, resuming the subsequent tasks also requires the ancestor failed tasks to restart at their adequate execution locations when failed. This is most of the time impossible in current systems. It requires repetitive incremental and partial checkpoints of state data and produced results, which can have an important overhead (Figure 7).

Secured Storage of Non-faulty Data. The secured storage of non-faulty data is essential for the optimization of the recovery process. Although, if it does not succeed, backtracking to a preceding checkpoint in the execution run is an option.

Restart Selection. There might be several options available for a single coordinated restart or local partial restarts (Figure 8). Depending on the situation, ranging from warnings to errors and fatal ones, the distributed configuration of the applications might render a global coordinated restart unrealistic. Several partial local restarts might be preferable, and in all cases, less expensive in terms of CPU and resource consumptions (Section “Coordinated Restart”, below).

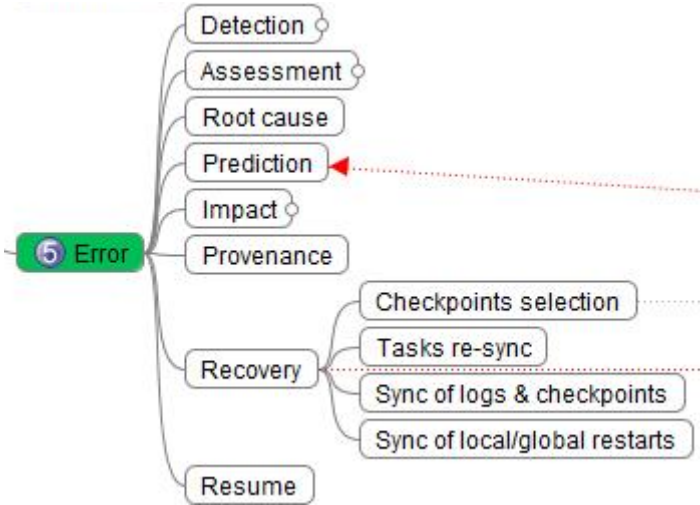


Fig. 7. Error recovery

Checkpoint Selection. An adequate checkpoint selection mechanism must be devised, which supports local restart in parallel and/or partial restarts from distributed and coordinated checkpoints. Here again, the versatility of the checkpointing mechanism, i.e., the support for multi-level checkpoints, is of first importance for reducing the restart overhead (Figure 9). But the cost is of course, the checkpointing overhead, both in terms of CPU and storage capacity, incurred. Encoding mechanisms, “shadowed” and “cloned” virtual disk images have been proposed to answer these concerns [23].

Coordinated Restart. Coordinated local restarts is a middle term option, between global cold restarts and multiple local restarts. As mentioned previously (Section 4.3), a global coordinated cold restart is unrealistic in distributed systems because it requires stopping all tasks and restarting the whole application, which might require large computing resources and days of CPU time. Coordination is fundamental here and related to distributed computations. It follows that coordinated restarts must be implemented by a specific mechanism that selects timestamped data and checkpoints. It also requires the careful selection of those checkpoints strictly needed for the restart. The latter will execute local restarts with the appropriate checkpoints selected.

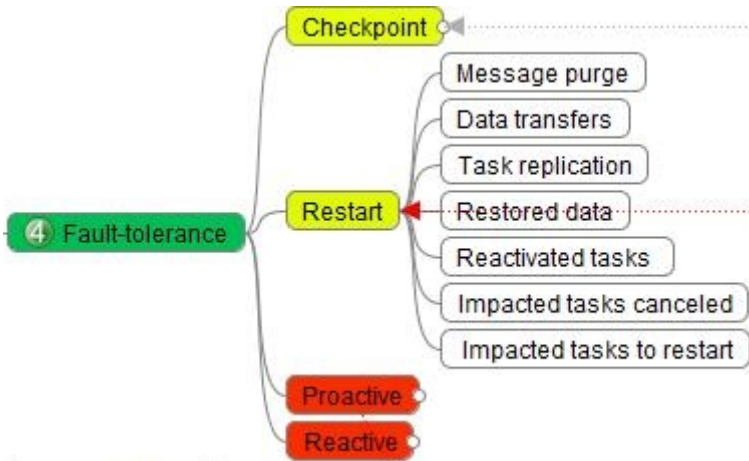


Fig. 8. Restart

5 Implementation

5.1 Overview

Several proposals have emerged recently dedicated to resilience and fault management in HPC systems [14][15][16].

The main components of such sub-systems are dedicated to the management of error, ranging from early error detections to error assessment, impact characterization, healing procedures concerning infected codes and data, choice of appropriate steps backwards and effective low overhead restart procedures.

General approaches which encompass all these aspects are proposed for Linux systems, e.g., CIFTS [5]. More dedicated proposals focus on multi-level checkpointing and restart procedures to cope with memory hierarchy (RAM, SSD, HDD), hybrid CPU-GPU hardware, multi-core hardware topology and data encoding to optimize the overhead of checkpointing strategies, e.g., FTI [22]. The goal is to design and implement low overhead, high frequency and compact checkpointing schemes.

Also, new approaches take benefit of virtualization technologies to optimize checkpointing mechanisms using virtual disks images on cloud computing infrastructures [23].

Two complementary aspects are considered:

- the detection and management of faults inherent to the hardware and software systems used
- the detection and management of faults emanating from the application code itself

Both aspects are different and imply different system components to react. However, unforeseen or incorrectly handled application errors may have undesirable effects on the execution of system components. The system and hardware fault management components might then have drastic procedure to confine the errors, which can lead to the application aborting. This is the case for out of bound parameter and data values, incorrect service invocations, if not correctly taken care of in the application codes.

This raises an important issue in algorithms design. Parallelization of numeric codes on HPC platforms is today taken into account in an expanding move towards petascale and future exascale computers. But so far, only limited algorithmic approaches take into account fault-tolerance from the start.

5.2 Resilience Sub-system

Generic system components have been designed and tested for fault-tolerance. They include fault-tolerance backpanes [5] and fault-tolerance interfaces [22]. Both target general procedures to cope with systematic monitoring of hardware, system and applications behaviors. Performance consideration limit the design options of such systems where incremental and multi-level checkpoints become the norm, in order to alleviate the overhead incurred by checkpoints storage and CPU usage. These can

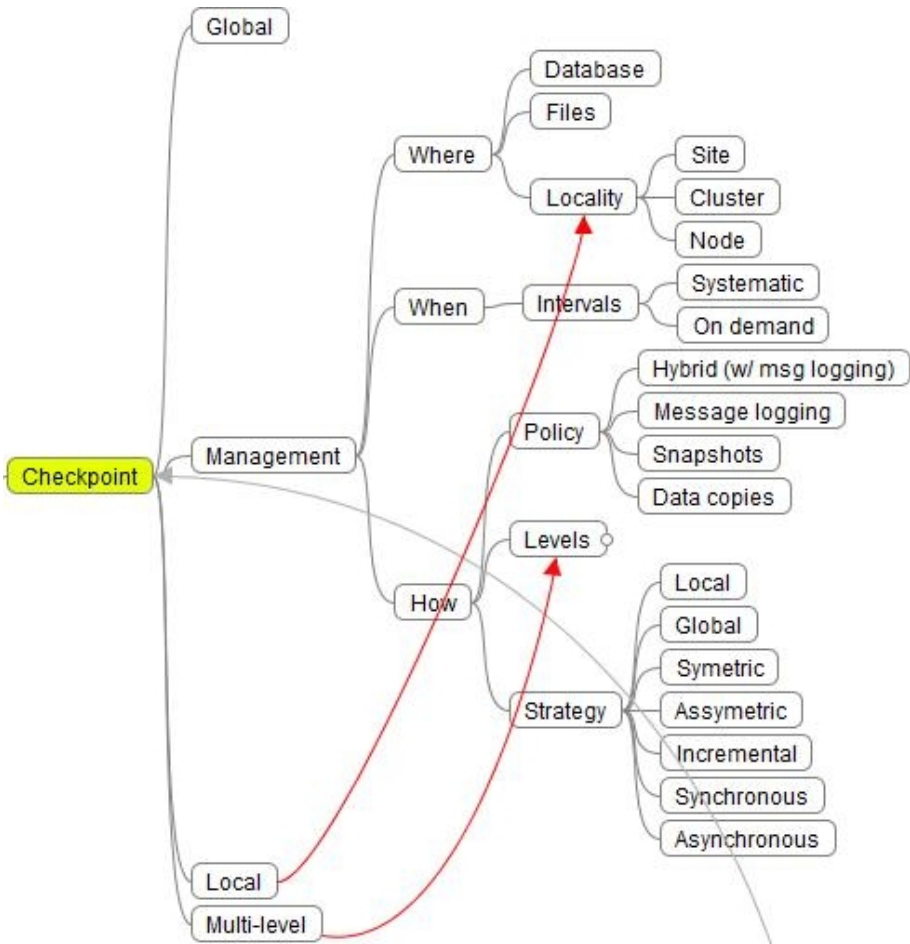


Fig. 9. Checkpoints

indeed exceed 25% of the total wall time requirements for scientific applications [22]. Other proposals take advantage of virtual machines technologies to optimize check-points storage using incremental (“shadowed” and “cloned”) virtual disks images on virtual machines snapshots [23].

6 Conclusion

The advent of petascale computers has raised concerns about system fault-tolerance and application resilience. Because exascale computers are now emerging, these concerns become even more stringent.

Sophisticated and optimized functionalities are therefore required in the upcoming hardware, systems and application codes to support effectively error management.

Large-scale applications also require distributed and heterogeneous environments to run collaborative multidiscipline projects. Workflow management systems are good candidate to deploy and control these applications because they require high-level and non-expert level dynamic features, e.g., interactive control and visualization. They also require dynamic reconfiguration capabilities, e.g., in case of task re-deployment because of hardware and software failures. Overall, they require resilience support because of potential software errors and erratic and unexpected behavior, e.g., because of wrong simulation parameters.

This paper defines concepts, details current problems and addresses solutions to application resilience. This approach is currently implemented and tested on simulation testcases using a distributed platform that operates a workflow management system interfaced with a cloud infrastructure. An automotive testcase is presented addressing a vehicle rear-mirror drag optimization.

The platform provides functionalities for application specification, deployment, execution and monitoring. It features resilience capabilities to handle runtime errors. It implements the cloud computing “Platform as a Service” paradigm while using a workflow system interface.

Acknowledgments. This work is supported by the European Commission FP7 Cooperation Program “Transport (incl. aeronautics)”, for the GRAIN Coordination and Support Action (“Greener Aeronautics International Networking”), grant ACS0-GA-2010-266184.

It is also supported by the French National Research Agency ANR (Agence Nationale de la Recherche) for the OMD2 project (Optimisation Multi-Discipline Distribuée), grant ANR-08-COSI-007, program COSINUS (Conception et Simulation).

References

1. Nguyễn, T., Trifan, L., Désidéri, J.-A.: A Distributed Workflow Platform for Simulation. In: 4th Intl. Conf. on Advanced Engineering Computing and Applications in Sciences, pp. 375–382. IARIA (2010)

2. Deelman, E., Gil, Y.: Managing Large-Scale Scientific Workflows in Distributed Environments: Experiences and Challenges. In: 2nd IEEE Intl. Conf. on e-Science and the Grid, pp. 165–172. IEEE, New York (2006)
3. Simon, H.: Future directions in High-Performance Computing 2009-2018. Lecture given at the ParCFD 2009 Conference (2009)
4. Dongarra, J., Beckman, P., et al.: The International Exascale Software Roadmap. *International Journal of High Performance Computer Applications* 25(1), 77–83 (2011), <http://www.exascale.org/>
5. Gupta, R., Beckman, P., et al.: CIFTS: a Coordinated Infrastructure for Fault-Tolerant Systems. In: 38th Intl. Conf. Parallel Processing Systems, pp. 145–156 (2009)
6. Abramson, D., Bethwaite, B., et al.: Embedding Optimization in Computational Science Workflows. *Journal of Computational Science* 1, 41–47 (2010)
7. Bachmann, A., Kunde, M., Seider, D., Schreiber, A.: Advances in Generalization and Decoupling of Software Parts in a Scientific Simulation Workflow System. In: 4th Intl. Conf. Advanced Engineering Computing and Applications in Sciences, pp. 247–258 (2010)
8. Joseph, E.C., et al.: A Strategic Agenda for European Leadership in Supercomputing: HPC 2020, IDC Final Report of the HPC Study for the DG Information Society of the EC (July 2010), <http://www.hpcuserforum.com/EU/>
9. Nguyễn, T., Trifan, L., Désidéri, J.-A.: A Workflow Platform for Simulation on Grids. In: 7th Intl. Conf. on Networking and Services (ICNS 2011), pp. 295–302 (2011)
10. Sindrilariu, E., Costan, A., Cristea, V.: Fault-Tolerance and Recovery in Grid Workflow Management Systems. In: Fourth International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), Krakow, Poland, February 15-18, pp. 475–480 (2010)
11. The Apache Foundation, <http://ode.apache.org/bpel-extensions.html#BPELExtensions-ActivityFailureandRecovery>
12. Beckman, P.: Facts and Speculations on Exascale: Revolution or Evolution? Keynote Lecture. In: 17th European Conf. Parallel and Distributed Computing (Euro-Par 2011), pp. 135–142. Springer, Heidelberg (2011)
13. Kovatch, P., Ezell, M., Braby, R.: The Malthusian Catastrophe Is Upon Us! Are the Largest HPC Machines Ever Up? In: Alexander, M., D’Ambra, P., Belloum, A., Bosilca, G., Cannataro, M., Danelutto, M., Di Martino, B., Gerndt, M., Jeannot, E., Namyst, R., Roman, J., Scott, S.L., Traff, J.L., Vallée, G., Weidendorfer, J. (eds.) Euro-Par 2011, Part II. LNCS, vol. 7156, pp. 211–220. Springer, Heidelberg (2012)
14. Riesen, R., Ferreira, K.B., Varela, M.R., Taufer, M., Rodrigues, A.: Simulating Application Resilience at Exascale. In: Alexander, M., D’Ambra, P., Belloum, A., Bosilca, G., Cannataro, M., Danelutto, M., Di Martino, B., Gerndt, M., Jeannot, E., Namyst, R., Roman, J., Scott, S.L., Traff, J.L., Vallée, G., Weidendorfer, J. (eds.) Euro-Par 2011, Part II. LNCS, vol. 7156, pp. 221–230. Springer, Heidelberg (2012)
15. Bridges, P.G., Hoemmen, M., Ferreira, K.B., Heroux, M.A., Soltero, P., Brightwell, R.: Cooperative Application/OS DRAM Fault Recovery. In: Alexander, M., D’Ambra, P., Belloum, A., Bosilca, G., Cannataro, M., Danelutto, M., Di Martino, B., Gerndt, M., Jeannot, E., Namyst, R., Roman, J., Scott, S.L., Traff, J.L., Vallée, G., Weidendorfer, J. (eds.) Euro-Par 2011, Part II. LNCS, vol. 7156, pp. 241–250. Springer, Heidelberg (2012)
16. Proc. 5th Workshop INRIA-Illinois Joint Laboratory on Petascale Computing, Grenoble (F) (June 2011), <http://jointlab.ncsa.illinois.edu/events/workshop5>

17. Capello, F.: Toward Exascale Resilience, Technical Report TR-JLPC-09-01. INRIA-Illinois Joint Laboratory on PetaScale Computing, Chicago (Il.) (2009), <http://jointlab.ncsa.illinois.edu/>
18. Moody, A., Bronevetsky, G., Mohror, K., de Supinski, B.: Design, Modeling and evaluation of a Scalable Multi-level checkpointing System. In: ACM/IEEE Intl. Conf. for High Performance Computing, Networking, Storage and Analysis (SC 2010), New Orleans (La.), pp. 73–86 (2010), <http://library-ext.llnl.gov>, also Tech. Report L LNL-TR-440491
19. Adams, M., ter Hofstede, A., La Rosa, M.: Open source software for workflow management: the case of YAWL. *IEEE Software* 28(3), 16–19, 211–219 (2011)
20. Russell, N., ter Hofstede, A.: Surmounting BPM challenges: the YAWL story, Special Issue Paper on Research and Development on Flexible Process Aware Information Systems. *Computer Science* 23(2), 67–79, 123–132 (2009)
21. Lachlan, A., van der Aalst, W., Dumas, M., ter Hofstede, A.: Dimensions of coupling in middleware. *Concurrency and Computation: Practice and Experience* 21(18), 75–82 (2009)
22. Bautista-Gomez, L., et al.: FTI: high-performance Fault Tolerance Interface for hybrid systems. In: ACM/IEEE Intl. Conf. for High Performance Computing, Networking, Storage and Analysis (SC 2011), Seattle (Wa.), pp. 239–248 (2011)
23. Nicolae, B., Cappello, F.: BlobCR: Efficient Checkpoint-Retart for HPC Applications on IaaS Clouds using Virtual Disk Image Snapshots. In: ACM/IEEE Intl. Conf. High Performance Computing, Networking, Storage and Analysis (SC 2011), Seattle (Wa.), pp. 145–156 (2011)

T-DMB Receiver Model for Emergency Alert Service

Seong-Geun Kwon¹, Suk-Hwan Lee², Eung-Joo Lee³, and Ki-Ryong Kwon^{4,*}

¹ Electronics Engineering, KyungIl University,
50, Gamasil-Gil, Hayang-Eup, Gyeongsan-Si, Gyeongbuk, Koea
seonggeunkwon@hanmail.net

² Department of Information Security, Tongmyong University,
535, Yongdang-dong, Nam-gu, Pusan, Republic of Korea
skylee@tu.ac.kr

³ Department of Information Communication, Tongmyong University,
535, Yongdang-dong, Nam-gu, Pusan, Republic of Korea
ejlee@tu.ac.kr

⁴ Department of IT Convergence and Application Engineering, Pukyong National University,
599-1, Daeyeon-3dong, Nam-gu, Pusan, Republic of Korea
krkwon@pknu.ac.kr

Abstract. This paper presents the method of emergency warning system operation based on T-DMB and the design of T-DMB AEAS(Automatic Emergency Alert Service) receiver model. The proposed receiver model compares the geographical location of emergency with the location of DMB transmitting station from T-DMB broadcasting signal and classifies the receiver location into alert region, neighboring region and non-alert region and transmits the emergency alert message according to each region. The geographical location of emergency can be obtained from FIG(Fast Information Group) 5/2 EWS(Emergency Warning Service) data field for AEAS message and the location of DMB transmitting station can be estimated from either the latitude and the longitude in main identifier and sub identifier in FIG 0/22 data filed for TII(Transmitter Identification Information) or TII distribution database. Thus, the proposed receiver model consists of the checking process of AEAS message from the received DMB signal, the verifying process of DMB transmitting location and the displaying process of AEAS message according to the classified region. In our experiment, we implemented the proposed receiver model with display section, storage section, DMB module for receiving broadcasting signal and control section and performed test emergency alert broadcasting using T-DMB signal generator. From experimental results, we verified that AEAS message can be displayed on the receiver that is located on alert region and neighboring region and cannot be displayed on the receiver that is located on non-alert region.

Keywords: T-DMB, Automatic Emergency Alert Service (AEAS), Transmitter Identification Information, Emergency Warning Service (EWS).

1 Introduction

Mobile phone has been the necessities of life in most people providing various service functions of wireless communication, game, scheduling, camera, multimedia playing

* Corresponding author.

and also DMB(Digital multimedia Broadcasting) service while preserving mobility. Therefore, mobile phone can be good medium for the alert information service. Wireless operators have provided the alert text messaging service to mobile phones in disaster area for mobile alert information service. But heavy traffic or interruption of wireless network create the delay of alert text messaging service. Furthermore, 3G mobile phone can't be provided this service model because CBS(Cell Broadcasting Service) providing alert text messaging was ruled out in 3G network service.

Recently AEAS(Automatic Emergency Alert Service) for T-DMB(Terrestrial Digital Multimedia Broadcasting) [1],[2] has been developed for solving some problems of alert information service and AEAS standard has been worked to recommendation of ITU-R BT.2049[3]-[5],[8]. T-DMB AEAS standard will be extended on the application domain of DMB technique and will be fundamental technique for creating a new mobile device market. This standard specifies definition, transmission and signaling of AEAS message and functional requirements of T-DMB AEAS transmitting system and AEAS receiver. This standard designs the format for AEAS message to be compact and essential information for fast transmission and selects FIG(Fast Information Group) 5/2 (EWS) in FIDC(Fast Information Channel Protocol) as transport protocol of AEAS message. But this T-DMB AEAS is the sub-channel message transport system and the public emergency alert service for all receivers that transmits AEAS message collectively without considering the relation of the alert region and the receiver position. Thus, AEAS message must be transmitted rapidly to only receivers in the alert region or the intended emergency area.

In this paper, we proposed T-DMB AEAS receiver model using TII(Transmitter Identification Information) [6],[7] that displays AEAS message according to the relation of the alert region and the receiver position, together with the method of emergency warning system operation based on AEAS standard. We consider two conditions for the proposed receiver model. The first is to keep up T-DMB and AEAS standards and satisfy service requirements for alert broadcasting that can activate automatically the receiver and does not interrupt regular program schedule. The second is to not only be serviced to all receivers but also display AEAS message to only receivers that are located at the intended emergency area. Considering two conditions, the proposed T-DMB AEAS receiver model consists of the EWS checking step, the extracting step of T-DMB transmitting station position and the AEAS message displaying step. The EWS checking step checks whether the emergency alert of EWS is announced from FIG 5/2 in T-DMB transmission frame. In case of EWS in FIG 5/2, the receiver model composes AEAS message from all FIGs and then process next two steps. The position extracting step extracts the geographical position of DMB transmitting station from TII signal symbol of FIG 0/22 in T-DMB transmission frame. Finally, the displaying step classifies the receiver position by the relation of the position of DMB transmitting station and the alert region and displays AEAS message according to the classified region. Therefore, the proposed receiver model can provide LAAS(Location Adaptive Alert Service), AAS(Automatic Alert Service) and NIAS(Non-interruptive Alert Service). We implemented the proposed receiver model with radio frequency section, audio section, display section, DMB module for receiving alert broadcasting and control section and evaluated the receiver performance from test alert broadcasting in a special region.

2 T-DMB Location AEAS Receiver Model

This paper presents the receiver model for T-DMB AEAS system with the method of emergency warning system operation based on T-DMB. In this section, we explain the emergency alert broadcasting process for the emergency warning system operation and then the T-DMB AEAS receiver model in detail.

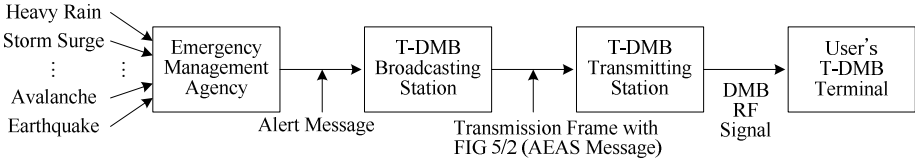


Fig. 1. The process of T-DMB Emergency Alert Broadcasting [6]

2.1 Emergency Alert Broadcasting Process

The generalized process of T-DMB emergency alert broadcasting consists of EMA(Emergency Management Agency or emergency announcement center), T-DMB broadcasting station, T-DMB transmitting station and mobile receiver as shown in figure 1. In this figure, EMA collects the information of emergency accidents and disasters and generates alert message and transmits it to T-DMB broadcasting station. This broadcasting station segments the received alert message to some FIGs for satisfying the AEAS message format and generates the transmission frame with FIG 5/2 EWS and transmits it to T-DMB transmitting station. Then the transmitting station inserts own geographical position into FIG 0/22 in the received transmission frame and transmits it to mobile receiver. Mobile receiver extracts TII signal in FIG 0/22 of the received transmission frame and identifies T-DMB transmitting station and display AEAS message according to the relation of the alert region and the position of T-DMB transmitting station.

EMA is the alert agency that provides AEAS message for the environment disaster with geographic information to T-DMB broadcasting station. It will be managed by country or city/province. The environment disaster contains the natural disaster of heavy rain, storm, earthquake, volcano, yellow sand and more or the human disaster of forest fire, traffic accident, building breakdown and more. The geographical information for the emergency alert region can be represented to codes that are allocated at cities or provinces in a country. Thus, after dividing a country into some administrative districts or geographical districts, each district is allocated to unique code with fixed bits. Its unique code for each district is used as the geographical information.

T-DMB broadcasting station composes transmission frame with ensemble programs and their associated information. If the alert message is received from EMA, this station segments the received alert message into some FIGs and encodes them to FIG 5/2 EWS of FIDC and inserts FIG 5/2 EWS into transmission frame.

T-DMB transmitting station receives transmission frame from T-DMB broadcasting station and converts the received transmission frame to DMB FR signal for

transmitting to mobile users. This transmitting station can be established in regular regional position in a country. T-DMB transmitting station in each regional position contains own latitude and longitude in LaC, LaF, LoC and LoF in main identifier and LaO, LoO in sub identifier in TII FIG 0/22 and assigns MainId and SubId to own code values in TII distribution table. Then this station transmits transmission frame with own geographical information to mobile users.

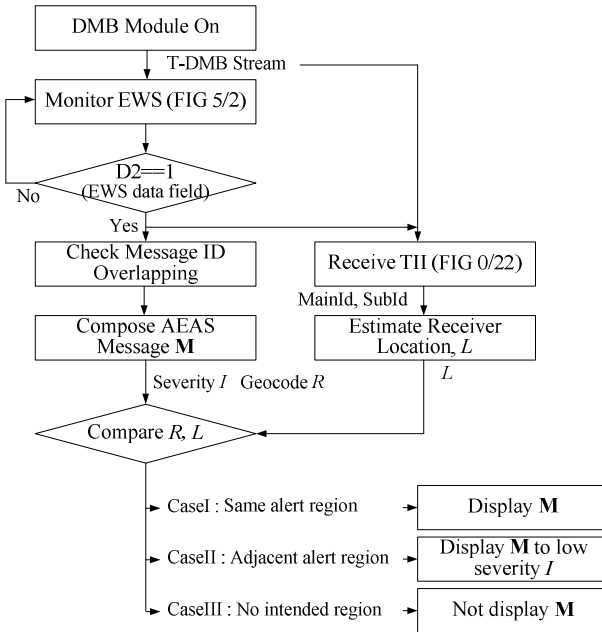


Fig. 2. The process of the proposed T-DMB AEAS receiver model

2.2 T-DMB AEAS Receiver Model

The receiver model estimates whether the receiver is in the alert region by comparing the position of T-DMB transmitting station and the alert region. Thus, if the transmitting position is within the alert region, the receiver model displays AEAS message. Or if the transmitting position is over the alert region, the receiver position divides the alert severity into warning alert or non-alert by the distance between two positions. The receiver model displays AEAS message with lower severity in case of warning alert and don't display AEAS message in case of non-alert. The process of mobile receiver model consists of the alert information checking step, the position extracting step of T-DMB transmitting station and the AEAS message displaying step, as shown in figure 2.

The receiver model powered on DMB module displays the tuned ensemble program by receiving DMB RF signal of the transmission frame from T-DMB transmitting station. DMB module transforms DMB RF signal to the signal that mobile device

can process and checks D2 value in FIG 5/2 whether EWS(Emergency Warning System) is present in the current transmission frame.

If D2=0 and EWS is not present, the receiver model displays the tuned ensemble program continuously. Or if D2=1 and EWS is present, the receiver model checks the overlapping of MessageId in FIG 5/2 EWS data field so that we know whether an AEAS message in EWS is overlapped or not. Thus, if the current MessageId is overlapped in the previous MessageId, the current AEAS message is the previous message. In this case, the receiver model displays the previous message continuously while receiving the transmission frame. If the current MessageId is not overlapped in the previous MessageId, a new AEAS message is being received. In this case, the receiver model completes a new AEAS message **M** from composing all message segments $m_i(i \in [0,15])$ in FIGs. Then from FIG 5/2 EWS data field, the receiver model extracts event code, severity p , geocode number N and geocode $R_k(k \in [1,N])$ an additional text information for displaying message. In general, AEAS message in the natural disasters will be announced in several regions since the natural disasters may be arisen in several regions. The receiver model stores Geocode number and several geocodes of alert regions for comparing the T-DMB transmitting position and them.

Table 1. TII distribution table for example [6]

District (Province/State)	City	Broadcasting station A		Broadcasting station A	
		MainId p	SubId c	MainId p	SubId c
P1 Province	C1 City	00	11	10	11
	C2 City	00	12	10	12
P2 Province	D1 City	01	11	11	11
	D2 City	01	12	11	12

Using the position of regional T-DMB transmitting station, we can estimate nearly the position where the receiver is located in current. After checking EWS flag and completing AEAS message, the receiver model extracts the geographical position of T-DMB transmitting station from TII signal of FIG 0/22 in transmission frame. This position can be extracted by two methods from TII signal of FIG 0/22. The receiver model process the extracting step of T-DMB transmitting station position only if EWS is present.

The first method is to extract the latitude TLa and the longitude TLo from main identifier and sub identifier in TII signal FIG 0/22. Thus, the latitude TLa and the longitude TLo with full precision of T-DMB transmitting station can be calculated by

$$TLa = LaC \times 90^\circ / 2^{15} + LaF \times 90^\circ / 2^{19} \pm LaO \times 180^\circ / 2^{19},$$

$$TLo = LoC \times 90^\circ / 2^{15} + LoF \times 90^\circ / 2^{19} \pm LoO \times 180^\circ / 2^{19}.$$

with latitude LaC and longitude LoC of coarse step, latitude LaF and longitude LoF of fine step and offset latitude LaO and offset longitude LoO . From the above equation, the receiver model extracts the geographical position $T=[TLa,TLo]$ of T-DMB transmitting station.

The second method is to extract the code value of MainId p and SubId c from TII distribution table. MainId p and SubId c represent to double figures $p=a_1a_2$ ($p \in [0,69]$) and $c=b_1b_2$ ($c \in [1,23]$). In a MainID p , the first figure a_1 indicates broadcasting station IDs and the second figure a_2 indicates administrative district IDs of each T-DMB broadcasting station, such as county, state or province. The double figures b_1b_2 of a SubId c indicate the regional T-DMB transmitting station that is located at city, village or town in administrative districts. Table 1 shows the example of TII distribution table. From this table, a_1 s are set to 0 and 1 for T-DMB broadcasting station A and B. a_2 s are set to 0 and 1 for P1 province and P2 province. b_1b_2 are set to 11 and 12 for two cities in each province. For example, a MainId $p=00$ and a SubId $c=11$ indicate T-DMB transmitting station that are located at C1 city in P1 province of T-DMB broadcasting station A.

In the message displaying step, the receiver model firstly compares the alert regions of geocodes in FIG 5/2 EWS and the geographical position of T-DMB transmitting station in TII signal FIG 0/22 and displays an AEAS message according to the distance relation of the alert region and the transmitting station position. The output of alarm or shake may be displayed together with an AEAS message. Thus, if the transmitting station position is within the alert region, the receiver model displays an AEAS message and keeps a user informed of an AEAS message using icon, alarm or shake. If the transmitting station position is neighbor on the alert region, the receiver model displays a message and an icon that informs a user that the emergency alert is announced in the alert regions. If the transmitting station position is far away the alert region, the receiver model do not display AEAS message and displays an alert icon so that a user see a message in later. The process for AEAS message displaying will be performed by two extracting methods of the transmitting position.

The display process is based on the latitude and the longitude $T=[TLa,TL0]$ of the position of the transmitting station as follows:

- 1) Verify the k th alert region $[RLa_k, RLo_k]$ of the latitude and the longitude from the geocode R_k ($k \in [1, N]$).
- 2) Check whether the position of the transmitting station $[TLa, TL0]$ is within the k th alert region $[RLa_k, RLo_k]$. If $[TLa \pm LaO, TL0 \pm LoO] \subset [RLa_k, RLo_k]$, classify the receiver position into the alert region and then go to step 4 for displaying the AEAS message. Alternatively, if $[TLa \pm LaO, TL0 \pm LoO] \not\subset [RLa_k, RLo_k]$, go to step 3.
- 3) If $[TLa \pm LaF, TL0 \pm LoF] \subset [RLa_k, RLo_k]$, classify the receiver position into the neighbouring region. If not, classify the receiver position into the non-alert region. Further, if k is N , then go to step 4. If not, go to step 1 after incrementing k by one.
- 4) If the receiver position is classified into the alert region, display the AEAS message along with an alarm, a shake, or an icon. Alternatively, if the position is classified into the neighbouring region, display a warning message and an icon that informs that an alert has been announced. Alternatively, if the position is classified into a non-alert region, display an alert icon instead of an AEAS message and a warning message.

The other display process that is based on the MainId p and the SubId c in the TII distribution table is as follows:

- 1) Verify the k th alert region from the geocode R_k ($k \in [1, N]$).
- 2) Check whether the alert region R_k is within the administrative district of the broadcasting station from the MainId $p = a_1a_2$. If R_k is within the administrative district, then go to step 3. If not, classify the receiver position into a non-alert region and go to step 1 after incrementing k by one.
- 3) If R_k is within the city of the transmitting station as determined from the SubId $c = b_1b_2$, classify the receiver position into the alert region and go to step 4 for displaying the AEAS message. If not, classify the receiver position into a neighbouring region and go to step 1 after incrementing k by one.
- 4) Display an alert message according to the classified region in the same manner as that of step 4 in the previous process.

3 Experimental Results

In our experiment, we implemented T-DMB AEAS receiver model using MainId and SubId of TII distribution table because it does not matter that the alert is announced by the city unit. And we performed the test emergency alert broadcasting for evaluating the receiver model. In this chapter, we explain the organization for implementing the receiver model and the result of test emergency alert broadcasting in detail.

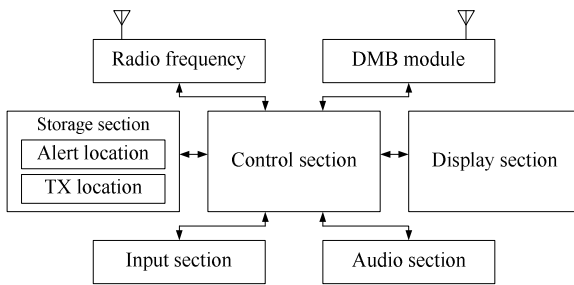


Fig. 3. The organization for implementing T-DMB AEAS receiver model [2],[6]

The organization of the proposed T-DMB AEAS receiver model consists of radio frequency section, input section, display section, audio processing section, storage section, DMB module of broadcasting receiver module and control section as shown in figure 3. The display section and the audio processing section are the output section that outputs an alert message and an alarm that are determined by the relation of the alert region and the transmitting position. The explanation of each section is as follows.

The radio frequency section sends or receives the signal associated with voice and data communication, SMS(Short Message Service) and MMS(Multimedia Message Service) by the control section. The input section contains several input keys and function keys for the information of number and character and the settings of various

functions. Thus, the input section generates keys that check the input signal for activating DMB module and the displayed AEAS message. The audio processing section contains a speaker for playing audio data and a microphone for collecting voice or other audio signals. This section output alarms that are predefined by the relation of the alert region and the transmitting position. For example, the receiver model in the neighboring region outputs a receiving alarm and an inform message that the alert regions are announced. Or the receiver model in the non-alert region a simple message sound instead of any alarms. The display section outputs a text of an AEAS message, an image or an icon to mobile LCD window. If any tuned ensemble program is being played in LCD window, an AEAS message will be overlaid on the playing program.

Our experiment used T-DMB/DAB+/DAB signal generator with ensemble generator, OFDM modulator and RF up-converter for T-DMB receiver development, as shown in figure 4. From this figure, the test AEAS message is generated to ETI(Ensemble Transport Interface) stream with FIG 5/2 EWS by the stream generator and this stream is converted to the T-DMB RF signal by ODFM modulator and RF up-converter. The T-DMB RF signal is transmitted to the T-DMB terminal through on-air analyzer. The binary code of the proposed receiver model is hard-ported to the T-DMB terminal. The T-DMB terminal displays an AEAS message by our receiver model algorithm.

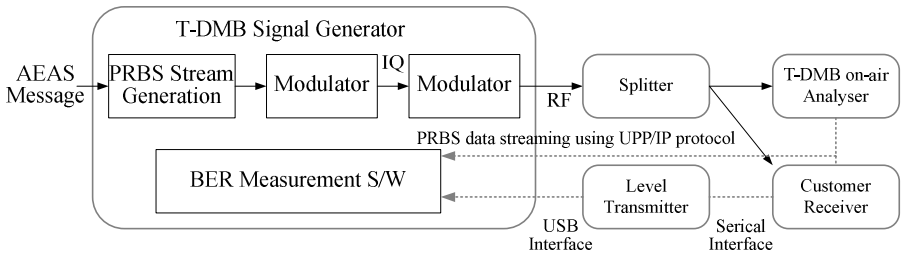


Fig. 4. Experiment test bed for T-DMB AEAS receiver model using T-DMB signal generator

Table 2. Test AEAS messages [6]

Section	Message 1	Message 2
Event type	CFW(Coastal flood warning)	CFA(Coastal flood watching)
Severity	Severe(Text+Alarm)	Moderate(Text)
Data&Time	15:44, Sept. 11, 2008	15:52, Sept. 11, 2008
Geocode number	1	1
Geocode	4913000000 (Jeju-Do, Segipo City)	4911000000 (Jeju-Do, Jeju City)
Text	Coastal flood warning driven by Tsunami in Segipo seashores. Look for a shelter from Tsunami urgently.	Coastal flood watching driven by Tsunami in Jeju seashores. Listen to news flash until it is cancelled.

We made two test AEAS messages according to event code, severity and geocode as shown in table 1 and converted them to the ETI stream with FIG 5/2 EWS. Thus, we announced a severe message 1 of CFW(Coastal flood warning) and a moderate message

2 of CFA(Coastal flood warning) to two cities in the same province. In our test broadcasting, EMA(emergency management agency) is used as the metropolitan city and the geocode of alert region is encoded to ASCII code of 10Byte. And the service ID in MCI signaling part is 0xF1E0024A and the service label is “AEAS”. We produced the stream of test emergency alert broadcasting in 12 minutes and announced a message 1 and a message 2 at once. The time schedule of stream is as follows.

- 1) 00:00, Start stream and send padding message of EWS (D2=0).
- 2) 05:00, Send AEAS message 1 (D2=1) and stop padding message.
- 3) 08:00, Send AEAS message 1 and AEAS message 2 simultaneously (D2=1)
- 4) 10:00, Send padding message (D2=0) and stop AEAS message 1 and AEAS message 2.
- 5) 12:00, End stream.

To test receiver model using the above test stream, we changed TII symbol of FIG 0/22 in TII distribution table while sending the fixed geocodes in test stream. Thus, we ported the changed TII symbol to T-DMB terminal every test measuring.

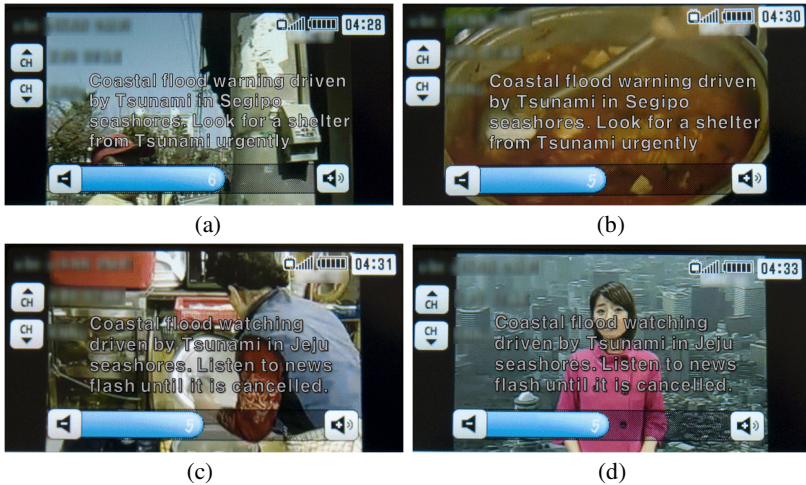


Fig. 5. AEAS messages displayed on DMB terminals in alert region

When hard-coded T-DMB terminal receiving TII symbol corresponding to Segipo city that is alert region of a severe message 1, the text in a severe message could be displayed on T-DMB terminal as shown in figure 5 (a) and (b). Also, when hard-coded T-DMB terminal receiving TII symbol corresponding to Jeju city that is alert region of a moderate message 2, the text in a moderate message could be displayed on T-DMB terminal as shown in figure 5 (c) and (d). When we changed TII symbol corresponding to another city in the same province and hard-ported it to T-DMB terminal, a warning message that informs of be announcing CFW and CFA to Segipo city and Jeju city respectively could be displayed on T-DMB terminal. In TII symbol corresponding to cities in other provinces, a simple icon that informs of transmitting an AEAS message in alert regions could be displayed on T-DMB terminal.

From the above experimental results, we confirmed that our T-DMB AEAS receiver model can provide all T-DMB terminals with EWS but only terminals that are located at alert region can display an AEAS message.

4 Conclusions

This paper presents T-DMB location AEAS receiver model using TII symbol. Our model compares the geocode of alert region in an AEAS message with the geographic location of Main/Sub identifier in TII symbol or TII distribution table of MainId/SubId and displays an AEAS message on only T-DMB terminals that are located at the alert region. Since our location AEAS model can provide all DMB terminals with LAAS(Location Adaptive Alert Service) based on transmitter, AAS(Automatic Alert Service) and NIAS(Non-interruptive Alert Service), it can be presented to the standard T-DMB terminal model for AEAS. Therefore, the proposed receiver model will provide the reliability service for the emergency warning system. An AEAS message may include main text as well as additional information of URI and multimedia in external link. In the future, we will develop T-DMB terminal model that can provides multimedia services for AEAS.

Acknowledgement. This research was supported by the Korea Research Foundation Grant funded by the Korean Government (MEST) (KRF-2011-0010902 and KRF-2011-0023118).

References

1. ETSI EN 300 401, v.1.4.1, Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers (January 2006)
2. Telecommunication Technology Association, Standard of Transmission and Reception for Digital Terrestrial Television Broadcasting, TTAK.KO-07.0014/R2 (2009)
3. Telecommunication Technology Association, Interface Standard for Terrestrial Digital Multimedia Broadcasting (T-DMB) Automatic Emergency Alert Service, TTAS.KO-07.0046/R2 (2009)
4. Choi, S.J.: Design of T-DMB Automatic Emergency Alert Service Standard: Part 1 Requirements Analysis. *Journal of Broadcast Engineering* 12(3), 230–241 (2007)
5. Choi, S.J.: Analysis of Emergency Alert Service and Systems. In: *International Conference on Convergence Information Technology*, pp. 657–662 (2007)
6. Kwon, S.-G., Jeon, H., Lee, S.-H., Kwon, K.-R.: Mobile Receiver Model for T-DMB Location Automatic Emergency Alert Service. *Journal of Korea Information and Communications Society (KICS)* 34(10), 727–813 (2009)
7. Kwon, S.-G., Jeon, H., Lee, S.-H., Kwon, K.-R.: TII based T-DMB location AEAS receiver model. In: *IEEE International Conference on Multimedia and Expo*, pp. 1286–1289 (2009)
8. ITU-R, Use of satellite and terrestrial broadcast infrastructures for public warning, disaster mitigation and relief, Recommendation ITU-R BT.1774-1 (2007)

A Framework for Context-Aware Systems in Mobile Devices

Eduardo Jorge¹, Matheus Farias², Rafael Carmo², and Wesley Vieira²

¹ Universidade do Estado da Bahia, Professor and Researcher, Bahia, Brazil
Instituto Recôncavo de Tecnologia, Researcher, Bahia, Brazil
emjorge1974@gmail.com

² Universidade do Estado da Bahia, Researcher, Bahia, Brazil
{matheusmf, rafael11jj, weslleyleandro}@gmail.com

Abstract. The pervasive and ubiquitous computing is a computing paradigm that aims to integrate the real world with the virtual world so that is not perceived by users. One of the areas of study of this paradigm is the context-aware systems that are systems that perform some action after collecting information that characterizes a given context. The objective of this paper is to describe the specification of a framework that can be utilized for the development of new context-sensitive applications.

Keywords: Ubiquitous Computing, Pervasive Computing, Mobile Computing, Android, iOS, Context of Use, CAMobile.

1 Introduction

An important area of research in the domain of mobile devices is what is being called the context of use. This concept aims to improve the user experience on mobile device.

The idea is to gather information, imperceptibly to the user, which characterize the current state of the same at the moment that the device is being utilized. By recognizing the context, the device must provide services that are of interest at that time for so improve the experience with the device.

One of the great needs of users is that the device could be adapted to various conditions of use, even when used by different users, for example, the user every time she goes practice running in a certain place, like to hear certain types of music. By detecting contextual information regarding the location of the user, the device automatically adapt its interface, so the songs that the user likes to hear are played on the player device.

To accomplish that objective, applications must make use of present sensors in mobile devices.

The growing evolution of the resources (such as increased processing capacity and storage, screen size, touch screen, insertion of GPS sensors, Bluetooth, accelerometer, camera, etc.) present in mobile devices, mainly smartphones and tablets, allowed

several applications emerge. Since mapping applications, audio and video playback, Internet access to business and educational applications.

At present, the simplest smartphone brings with multiple applications, each designed for a specific purpose, using one or more sensors of the device. At this point, is a major bottleneck. The various applications installed on the device make use of sensors many times simultaneously. Thus, each application needs to receive information from a particular sensor, for example GPS, would create an "instance" of this sensor by overloading the operating system by having to manage multiple identical copies of the same information, and increasing consumption of resources, such as processing, battery, memory etc..

Another factor that hinders the development of applications that use context of use, is that the developer than to worry about the logic of your application, also needs to worry about how to retrieve the information you need for each sensor.

The aim of this paper is to show the specification of CAMobile (Context-Aware Mobile), a computational solution where different applications can be build taking advantage of a common architecture, simplifying the development of context-sensitive systems for mobile devices.

The use of an architecture like that will be specified, has some advantages, for example: Reuse of code, since the code that handles certain sensors would be repeated in various applications, Reduction of complexity, because the developer need only be concerned with logic of their application; Reduced battery consumption, as the proposed framework provides centralized access to sensor device.

Some applications using the CAMobile are planned to be developed to validate the architecture specified. In section 7 of this paper is exemplified some of these applications.

This article is divided into seven main topics. The first refers to theoretical information about Pervasive and Ubiquitous Computing and Context-Aware Computing.

In section 4, CAMobile is specified in a generic form, explaining each of its components and their pattern of communication with applications. Section 5.1 explained the main concepts of the Android platform, while in section 5.2 is shown adaptation of CAMobile in this platform. Is discussed in section 6.1 the main concepts of iOS platform, while in section 6.1 is shown adaptation of CAMobile in this platform.

Section 7 gives some scenarios and applications where CAMobile can be used. Finally in section 8 is made the conclusion of the article.

2 Ubiquitous and Pervasive Computing

The term ubiquitous computing was first used by [12], to describe a vision of the future where computers would be ubiquitous, that is, would be inserted in various objects: watches, pens, light switches, cups, etc. and everywhere: in the city, in the forest, beach, street, etc.

In this view, the various computing devices (computers physically connected to the network, mobile devices and objects as mentioned above) would be scattered and

fully integrated into the environment and on the lives of users as to not be noticeable for its performance and its use is as natural to the point of becoming invisible to users.

There are computers, even in a simple form, on cell phones, microwaves, refrigerators, ATMs, etc.. However, for [2], not enough simply to provide user interaction at all times and everywhere. It is necessary in this model of computation, that applications adapt to this scenario highly distributed, heterogeneous, dynamic and mobile, thus providing the information the user needs for a given context. It is this scenario that the context-sensitive computing is inserted.

3 Context Sensitive Computing

One of the main areas of research in computing ubiquitous is computing context-sensitive. This paradigm seeks to develop applications that take advantage of context information at the time of use of computing devices. The first definition of the term context-aware computing has been defined as "the study of applications that adapt according to user location, group of persons, objects near the user and the changes occurred to those objects along the time," by [11].

The definitions for the term context-sensitive computing found in literature closely resemble the above mentioned: [10]: Applications that dynamically modify or adapt their behavior based on information in the context of the application or user, [4]: A system that uses information relating to the context to provide information or services relevant to the user.

The main purpose of context-sensitive computing is to collect for computing devices (objects cited in ubiquitous computing: watches, pens, computers, cell phones or any device that has the ability to communicate with others or with the internet) inputs (information) to enable them to characterize the conditions of a user, the environment in which he is inserted at the moment, the devices around them, etc..

This information, called "context" was defined by [3] as: "Any information that could be used to characterize the situation of an entity. An entity can be a person, place or object considered relevant to the interaction between a user and an application, including the user and the application." That is, context refers to information relevant to a user at a given moment, such as his location, environment in which it is inserted, the device processing power, energy, memory, nearby devices, time of day, among others.

The context-sensitive computing is strongly linked to mobile devices, because of mobility of the users and computational objects and the fact that the user desires information anywhere at any time, especially in movement, making the user context is in constantly changing (the user moves from one place to another, thus modifying the conditions of the environment, other users and / or resources to move in and out of the application area of interest, restriction of resources available to execution of the application) [8].

4 Framework

This section is specified the CAMobile and shown your workflow.

4.1 CAMobile

The framework CAMobile is divided into three components as presented in Figure 1.

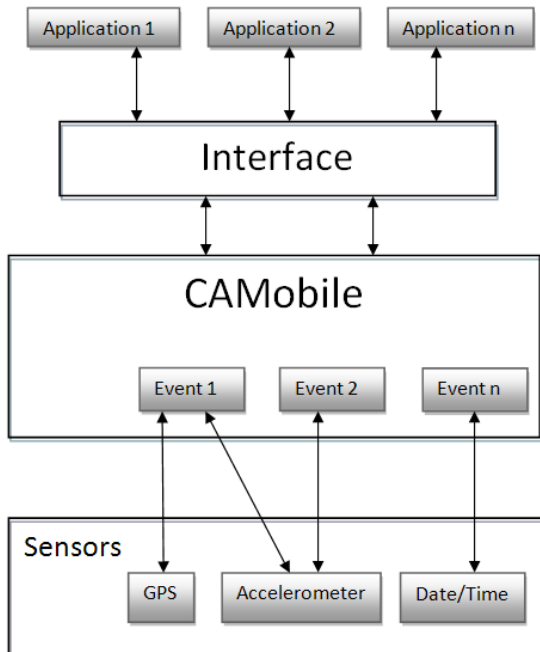


Fig. 1. High-level architecture

The first component called Interface has as main function, to provide for external applications a set of classes and interfaces that standardize the communication with the CAMobile. In addition, it also informs what events are available in CAMobile to be observed and what the possible conditions of notification for each event. This library should be imported for each application that want to connect with the CAMobile.

The second and main component is the framework CAMobile itself. It is responsible for meeting and managing the requests made by applications and observes the events necessary for can respond to applications. The CAMobile observes the existing events, following a definite pattern, inspired by the Observer design pattern, where the CAMobile is notified of the change from the various sensors. It is also responsible for identifying when events should be initiated and finalized.

The third component, called "Events" is small pieces of code responsible for "observe" the sensors of the device (camera, clock, accelerometer, gyroscope, etc.) and notify CAMobile when conditions of sensors attend your request. Each event is implemented to observe a single sensor and its main objective is to isolate the other components of the access to the sensor, assuming the role of interface between them. Several events can be coupled to CAMobile, each responsible for monitoring a sensor device, being limited only by available hardware resources on the device.

4.2 Flow Operation of CAMobile

When an application connect with CAMobile, and require the observation of an event, need to follow a standard communication defined by the component Interface. This standard defines the following information must be passed by the applications:

Event: Code known for applications that uniquely identifies that event;

Notification condition: It is the information on which the application is interested. The framework will start observing the specified event and when the context coincides with the conditions provided, the application will be notified.

Each event has its own conditions of notification. Some examples are: the event GPS may have the condition to notify the application when the device is within 10 meters of a certain place, and the event Bluetooth can notify the application when the device is close to another defined device.

Observer: It informs who should be notified when the conditions are met.

After identifying the requested events, the CAMobile will start just those who are not already running, being responsible for finalizing them when there are more applications interested in them. In this way, there is noun necessary power consumption.

Once the events were initiated and are observing the sensors of the device, the CAMobile will wait new requests from external applications or notifications of events when conditions are satisfactory. After being notified by the events, the Service will notify interested applications.

An application may be interested in more than one event. In this particular case, from the aggregation of two or more simple events, brings up a combined event. This aggregation is nothing more than a logical combination of the conditions for each event using the clauses "and" or "or." For example, an application requests the event date / time and GPS, and wants to be notified all days, 08:00, if the device is in a certain location.

When the CAMobile receive this request, it will instantiate the events of date / time and GPS, if not already instantiated, and only notify the application when the two conditions (when it is 08:00 and at the specified location) are satisfied, if the combination logic used is "and". Or notify the application so that one of the conditions (or when it is 08:00 or when at the specified location) is satisfied, if the combination is logical "or". The application shall inform the logical condition should be used when more than one event is observed.

5 Definition of the Framework in the Android Platform

This section contains the theoretical foundations of the Android platform and high-level description of a framework for context-aware systems in this platform.

5.1 Android Platform

The Android, released in October of 2008, is a development platform for mobile applications whose operating system is based on Linux. Its appearance is a result of the union of several cellular phone companies led by Google called the Open Handset Alliance (OHA). This group currently is formed by 84 members, among which are: LG, Motorola, Samsung, Sony Ericsson, etc..

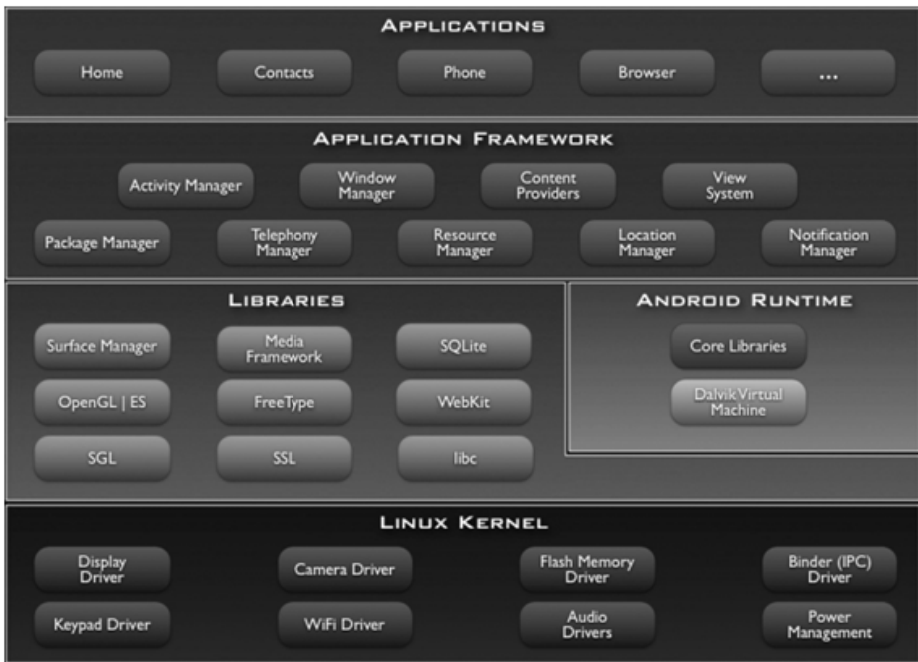


Fig. 2. The main components of the Android (Source: <http://developer.android.com/guide/basics/what-is-android.html>)

Applications: They are native Android applications like email client, browser, maps, etc. All applications are done in Java.

Application Framework: Layer that offers a collection of frameworks with several features of free access to applications. Are classes for building interface, data access of the device and of the memory card, among others.

Libraries: Libraries in C / C ++ used by various components of the Android. The functionalities of these libraries are available to developers by the Application Framework layer.

Android Runtime: Has a set of libraries with most of the functionality of the Java language and a virtual machine optimized for devices with limited resources.

Linux Kernel: Abstraction layer between the hardware and the rest of the software stack.

5.2 Framework in Android

In Android, CAMobile can be initiated in two ways: automatically, immediately after the device is switched on, or by an external application. Only one instance of the framework will be running to suit all applications, aiming at energy saving device. If another application has an interest in CAMobile and this is already running, it will simply bind to it. The Service class of the operating system, as shown in Figure 3, guarantees this behavior.

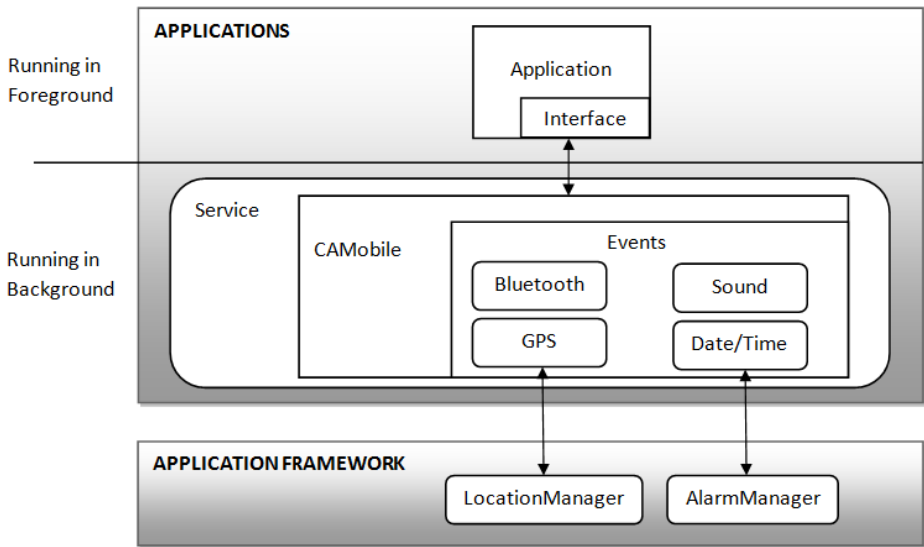


Fig. 3. Architecture Framework on Android

Because it is a free processing of user interaction, the service will run in the background, in addition to being independent of any application. This way, even if the application that has begun will be finalized, the service continues active to answer other applications.

In Android, the lifetime of a service is controlled by the operating system and has highest priority over any other process running in the background. This will only be terminated in critical condition of memory and resources.

Although Android has terminated service CAMobile by limited resources, the operating system itself will attempt to restart it when conditions are favorable.

However, the events of the framework that will observe the sensors device will use Android's native APIs to attend requests from external applications (figure 3), such as LocationManager for geo-information, or the AlarmManager for schedule tasks.

6 Definition of the Framework in the iOS Platform

Here, theoretical fundamentals of the iOS platform and high-level description of a framework for context-aware systems for this platform.

6.1 iOS Platform

The IOS is the operating system base on UNIX that runs on devices such as iPhone, iPod Touch and iPad. This system controls the hardware devices and provides the technologies needed to implement native applications.

The iOS's architecture is similar to basic architecture of Mac OS X. So, the system acts as an intermediary between the hardware and applications. The architecture is divided into layers, where the lowest level is the services and technologies essential for all applications, while at the highest level is more sophisticated services and technologies

Figure 4, shows the layers of the IOS.



Fig. 4. Layers of iOS Platform (Source: <http://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Art/SystemLayers.jpg>)

The Cocoa Touch layer takes care of the visible part of the user that is the interface that will appear on the screen. It also controls the user interaction with the screen through touch, the accelerometer or the use of cameras.

This layer is the most used by the developer because in it is the main frameworks that are used in the building of applications, giving supporting for technologies such as multi-tasking, touch-based input, push notification and other high-level system services.

The Media layer is responsible for the multimedia resources of the devices, such as sounds, images and videos that are played.

The Core Services layer manages the services of the device, as the phone calls, the sending of text message, services of protocols network communications and of the database.

The Core OS layer is the lowest level of the system and the most basic. It controls the functioning of the operating system, managing memory, battery, screen brightness, security, among other functions.

6.2 Framework on iOS

The idea of a service running in the background is not adequate for the iOS platform, this is because the concept of multitasking is a bit different in this platform.

Table 1 contains the Apple description of the states of a application in this platform.

Table 1. States of applications on iOS

State	Description
Not running	The app has not been launched or was running but was terminated by the system.
Inactive	The app is running in the foreground but is currently not receiving events. (It may be executing other code though.) An app usually stays in this state only briefly as it transitions to a different state.
Active	The app is running in the foreground and is receiving events. This is the normal mode for foreground apps.
Background	The app is in the background and executing code. Most apps enter this state briefly on their way to being suspended. However, an app that requests extra execution time may remain in this state for a period of time. In addition, an app being launched directly into the background enters this state instead of the inactive state.
Suspended	The app is in the background but is not executing code. The system moves apps to this state automatically and does not notify them before doing so. While suspended, an app remains in memory but does not execute any code. When a low-memory condition occurs, the system may purge suspended apps without notice to make more space for the foreground app.

When an application in Active state is interrupted due to a phone call or because the user pressed the key 'Home' of the device, the application switches to Background state and has only five seconds to save data and information of the interface for when the application return to Active state everything is as before. Then, the state goes to Suspended.

However, it is possible that an application asks for more time to perform some tasks. But, this execution could take no more than ten minutes, after which time the operating system changes the state to Suspended.

In addition, there are five types of applications that can stay running in the background for a long period of time without the operating system suspends. Table 2 has the Apple description of these types.

Table 2. Types of applications that are allowed to run for a long time in the background

Type	Description
audio	The app plays audible content to the user while in the background.
location	The app keeps users informed of their location, even while it is running in the background.
voip	The app provides the ability for the user to make phone calls using an Internet connection.
newsstand	The app is a Newsstand app that downloads and processes magazine or newspaper content in the background.
external-accessory	The app works with a hardware accessory that needs to deliver updates on a regular schedule.

It is necessary notify the iOS that the application belongs to these types through the declaration of the variable 'UIBackgroundModes' in file 'Info.plist'.

As one service that stay monitoring the existence of contexts of use requires more than ten minutes executing in the background and does not fit into any of the types allowed to run over a long period of time is inappropriate to create a service of this type for this platform.

With this, was taken the service that had in the specification of of the framework for the Android platform. Figure 5 shows the final iOS's architecture of the framework for the iOS platform. By removing the service, which would work as a unique instance, would lose the advantage of economy of resources and bacteria. However, the iOS already managing some of its sensors, and thus, there not is "waste" of resources.

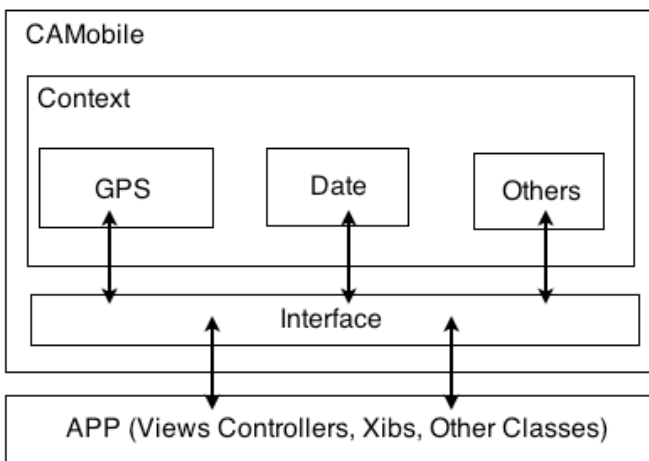


Fig. 5. Framework Architecture in iOS

In the figure 5 shown that an application consists of classes, controllers, xibs (and other files that comprise it) and by the framework CAMobile, which have some boxes that are the contexts of use like location and time.

The CAMobile act as an interface between the application classes and the sensors of the devices that bring context information such as GPS, clock (date / time) and among others.

Each box that represents a context of use will use frameworks of the iOS to obtain information of the context of use. For example, the GPS box will use the CoreLocation Framework to get location information such as latitude and longitude.

The use of sensors that provide information of context of use will be simplified to a developer of context-sensitive systems. This is because the CAMobile will have methods that will facilitate the verification of certain contexts of use. It can even, leave the application scheduled to an event be executed when one or more contexts of use (together) is found.

7 Scenarios and Applications

This section will be exemplified four scenarios where it could build applications that make use of the framework.

7.1 Shopping Promotions

Products on offer are a store sales grow, because people tend to take advantage of promotions in order to save.

When a person goes into a mall, many times has no knowledge of promotions exist in the day. This happens because the mall has many stores and people do not always have enough time to visit them all.

This way, an application can be made that aims to inform users about the existence of promotions inside a mall. When the user enters the mall, the application would perceive it by GPS and then check on a server if a store has a promotion available. If any, a notice is posted to the user asking if he want to view the promotions.

Through this application a mall can negotiate with the owners of the stores the announcement of promotions and advertising space.

7.2 Mobile Devices in Traffic

When driving a vehicle, there may be cases where the driver needs to interact with your mobile device, either to answer or make a call, to locate in the GPS or move to your favorite music. However, for security reasons, the driver's attention should be focused on traffic, it is necessary that the driver be able to perform their task with minimal interaction. To do this, applications to improve the interface of the device for fast and easy to use are recommended.

However, enable them every time that the driver is driving is a repetitive process and can be easily forgotten. The ideal would be that the device automatically recognize the moment when the user is driving.

This is possible through a context application. With it, the device would be able to identify the speed with which it is moving and could infer whether the user is driving or not. With this context recognized the device could automatically change the interface, freeing the user from the obligation to change it manually every time that he will drive.

7.3 Photos around the World

It is very common during a vacation people go to various places and take pictures with his cell phones to keep memories of them.

However, in reviewing your photo album, after some time, people can end up forgetting the exact context in which picture was taken (place, date, time, etc.).

A solution to this problem would be the development of an application that would give context information for each new captured picture. For this, the application should be aware of variations in the data store of the device, so that when a new photo was added, it automatically collects information from the context and bind to the newly captured photo.

With this information, the application could offer to the user an alternative view of the photos present in the cell. By means of a map, a person could observe exactly the path taken during the trip and where the photos were taken.

7.4 Parking Locator

It can be seen in recent years, a significant increase in the number of vehicles on city streets. But the infrastructure of cities did not have followed, creating a major problem for drivers: Where to find a parking lot near where he meets with available vacancies?

To solve this problem, would be developed a context-sensitive application that would inform the user the parking lots that are close to him at that moment, indicating whether they have vacancies. The application must observe changes in user's position through the GPS sensor of the device, and each change, check whether the new user's position is close to some previously registered parking lot, in order to display this information to the user.

Thus, the user, via a map, could see all the parking spaces nearby and which ones have vacancies available for parking.

8 Conclusion

This section presents the final considerations of the subject of the article.

The context-sensitive computing is one of the main areas of research in ubiquitous computing. It aims to develop applications that use context information (for example, location, time, objects), when the use of computing devices. By obtaining this information the applications will execute actions imperceptibly for the user, so that the user experience with the device becomes nice.

For a developer to build applications of this type he needs to know how to handle presents sensors in devices such as GPS and accelerometer, something that is not trivial. To simplify this, was specified in this article an architecture of a framework called CAMobile, which encapsulates the handling of sensors of the devices and to provide a communication interface with it. Thus enabling, to schedule an event to be triggered after being satisfied (detected) the conditions or a set of points (contexts, for example, location and time) in a given period.

The architecture of CAMobile can be used on any mobile device platform. Can be adapted depending of the advantages and limitations of each platform. That adaptation was shown to Android and iOS.

In addition, the reuse of the specified architecture provide benefits such as code reuse, reduced complexity and reduced resource consumption of the device.

References

1. Android developer, What is Android?, <http://developer.android.com/guide/basics/what-is-android.html>
2. Barbosa, J., Hahn, R., Rabello, S., Pinto, S.C.C.S., Barbosa, D.N.F.: Computação Móvel e Ubíqua no Contexto de uma Graduação de Referência. *Revista Brasileira de Informática na Educação* 15(3), 53–65 (2007)
3. Dey, A.K.: Understanding and Using Context. *Personal Ubiquitous Computing* 5(1), 4–7 (2001)
4. Dey, A., Abowd, G.: Towards a Better Understanding of Context and Context Awareness. In: *Proceedings of CHI 2000* (2000)
5. iOS Developer Library. iOS App Programming Guide, http://developer.apple.com/library/ios/#documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/ManagingYourApplicationsFlow/ManagingYourApplicationsFlow.html#//apple_ref/doc/uid/TP4007072-CH4-SW3
6. iOS Developer Library. iOS Technology Overview, http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007898
7. Lecheda, R.: *Google Android – Aprenda a criar aplicações para dispositivos móveis com Android SDK*, 2ª Edição, Novatec Editora (2010)
8. Loureiro, A.A.F., Oliveira, R.A.R., Silva, T.R.M.B., Júnior, W.R.P., Oliveira, L.B.R., Moreira, R.A., Siqueira, R.G., Rocha, B.P.S., Ruiz, L.B.: *Computação Ubíqua Ciente de Contexto: Desafios e Tendências*. In: *Simpósio Brasileiro De Redes De Computadores E Sistemas Distribuídos*, Recife, vol. 27, pp. 99–149 (2009)
9. Open Handset Alliance, <http://www.openhandsetalliance.com/>
10. Ryan, N., Pascoe, J., Morse, D.: Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant. In: Gaffney, M., et al. (eds.) *Computer Applications in Archeology*. Tempus, Oxford (1997)
11. Schilit, B., Theimer, M.: Disseminating Active Map Information to Mobile Hosts. *IEEE NetWork* 8(5), 22–32 (1994)
12. Weiser, M.: The computer of the 21st Century. *Scientific American* 265(3), 66–75 (1991)

A Simulation Framework for Scheduling Performance Evaluation on CPU-GPU Heterogeneous System

Flavio Vella¹, Igor Neri^{2,3}, Osvaldo Gervasi¹, and Sergio Tasso¹

¹ University of Perugia, Dept. of Mathematics and Computer Science, Perugia, Italy

² University of Perugia, Dept. of Physics, Perugia, Italy

³ University of Perugia, Dept. of Electronic and Information Engineering,
Perugia, Italy

Abstract. Modern PCs are equipped with multi-many core capabilities which enhance their computational power and address important issues related to the efficiency of the scheduling processes of the modern operating system in such hybrid architectures.

The aim of our work is to implement a simulation framework devoted to the study of the scheduling process in hybrid systems in order to improve the system performance. Through the simulator we are able to model events and to evaluate the scheduling policy for heterogeneous systems. We implemented as a use case a simple scheduling discipline, a non-preemptive priority queue.

Keywords: Simulation, Scheduling, Heterogeneous Systems, GPU, OpenCL, Operating System.

1 Introduction

High Performance Computing with hybrid or heterogeneous systems attracted a lot of interest in academia and research in the last years. This is most due to the ubiquity of graphical processing units (GPU) in modern PCs which, thank to the development of specific framework like CUDA [1] or more recently OpenCL [2], enables the user to use high performance scalar processors along traditional computational units. In addition, the modern PCs are often enabled with several multi-core processors and an efficient scheduling among the various available resources is an important issue to be addressed.

A system composed by computing units, with different characteristics and strategies for data processing is usually called hybrid system (\mathcal{H} -system) due to the presence of heterogeneous computing units. Usually the allocation of the computing resource (especially on the GPU side) is demanded to the user, who decide which device to use and when. This is resulting in an inappropriate or inefficient scheduling of jobs and processes and to an unoptimized use of the hardware resources.

Some works has been made to try to integrate CPU and GPU resources in batch system[3], trying to allocate in an appropriate way the computational resources based on job or user requirements. It is worth to be noticed that in the modern operating systems the optimization of the process execution in \mathcal{H} -systems it is not well addressed.

In this work we propose an \mathcal{H} -system simulator to test scheduling algorithms for hybrid computing system. The simulator enables the user to write his own scheduling algorithm, evaluating it in a \mathcal{H} -system. Furthermore the simulator enables the user to select the specific distribution of submitted processes and the system configuration in terms of CPU and GPU resources. The output information provides a full report related to the scheduling process, including the statistics of the mean waiting queue time, the involved devices and the completion time.

Our paper is organized as follows: the related works is described in section 2; some consideration about programs, processes and threads are presented in section 3; the simulator structure and implementation details are defined in section 4; and the concluding remarks and future directions are provided in section 5.

2 Related Work

The main purpose of a scheduling algorithm is to establish the access method to system resources by specific jobs. In particular, the Operating System scheduler enables the multitasking kernels distributing processes and threads on available resources on the basis of their priority. Recently many works have been made to study and improve the scheduling performance. In systems with homogeneous computing elements there are several consolidated scheduling algorithms like multilevel feedback queue in Windows and Mac OS or Completely Fair Scheduler (CFS)[4] in recent versions of Linux. Some efforts have been made to try to design efficient scheduling algorithms also for an \mathcal{H} -system (CPU-GPU) like in [5], [6] and [7].

Scheduling algorithms need to be tested in different environments and conditions, sometimes analytical model can be used to analyze and validate scheduling efficiency in specific configuration, however a simulation infrastructure is often needed to perform tests. In [8], the authors present an high performance simulator for a Grid environment; the efficient scheduling of heterogeneous computational resources is a cool issue for Grid environments too. In the paper they use a discrete event simulator adding blocks for grid scheduling. Similar works are presented in [9] and [10].

Our research scope is to develop an event based simulator to test scheduling algorithms for hybrid or heterogeneous systems. The presented simulator is able to evaluate the performance of an \mathcal{H} -system scheduler based on classic evaluation metrics [11] like minimization of queuing time and execution time and maximizing the use of the available resources.

3 Program, Processes and Threads

A process is an instance of a program, each process has the ability to keep the state of the program during its execution like variable, hardware state and the content of an address space in memory. A process can be divided in multiple threads, each thread share the same address space of memory with other threads of the same process. Obviously only one thread can be executed on a CPU at a given time. Each thread is usually specialized and tends to be CPU-bound or I/O-bound, quite remarkably the first type spend a lot of time using the CPU and the other one waiting for I/O operations to complete (keyboard interrupt, audio and so on). Many schedulers do care about whether or not a thread should be considered CPU or I/O bound, and thus techniques for classifying threads represent a relevant scheduler's component. Some schedulers tend to give CPU access priority to I/O bound threads, because I/O operations take long time it is good to start relative threads earlier.

In hybrid systems (CPU/GPU) a classifier able to decide the type of threads (CPU or I/O bound) can also be used to determine in which device the thread has to be executed. To understand the work-flow of the processes in a hybrid system it is important to provide a more detailed description of how GPU threads are executed. The details of process execution depend on the architecture used and how heterogeneous computing units interact. In this work we consider the processes work-flow as defined by OpenCL.

OpenCL is a standard, supported by the Khronos Group, for parallel cross-platform programming of modern processors such as GPUs and many-core CPUs. The execution of OpenCL applications can be divided in two parts: host program and kernel program. The host program, which is executed on CPU, defines the context (platform layer) for the kernels and manages their execution. An instance of the kernel program is executed on each point in the index space, this means that multiple instance will be executed in parallel. It is important to highlight that for each execution of a process on the GPU a program is executed on CPU, meaning that the scheduler has to manage a CPU thread while the GPU is occupied. Furthermore each execution on GPU is composed by multiple threads that run in parallel. Since the execution of those threads is simultaneous, we can treat it as a single thread execution. In the next sections we will use the term job to refer both to single CPU thread or parallel job on GPU composed by several threads. According to [12][13] the jobs have an inter-arrival time exponentially distributed and an execution time is chosen according to a Gaussian distribution. The average execution time and its standard deviation depend on the nature of the type of job, in particular if it is real time (RT) or not (in the latter case we name it User Job).

4 HPSSim

In this section, a simulation model for CPU-GPU on a \mathcal{H} -system is defined and described. The proposed model aims to simulate a \mathcal{H} -system composed by

a set of processors (CPUs) and graphics cards (GPUs) used as compute units to execute heterogeneous jobs, a classifier selecting the type of compute device (CPU or GPU) and a scheduler which implements the policy to be evaluated.

The proposed CPU-GPU simulation model is defined in terms of a set of state variables describing the system (such as devices, jobs, queue and scheduler), and a state transition function which determines its progression through a finite set of discrete events.

The model is formally defined as a tuple:

$$\mathcal{H}_{sm} = (\mathcal{S}, \mathcal{D}, \mathcal{J}, \mathcal{E}, \delta)$$

where \mathcal{S} represents the queue model which will be defined in Section 4.2; \mathcal{D} and \mathcal{J} represent, respectively, the system device collection and the set of jobs which will be processed. Additional details will be provided in Section 4.1. \mathcal{E} is the events set generated and run through the simulator, defined in Section 4.4. Finally, δ represents the transition states function, defined as $\delta : (\mathcal{Q}, \mathcal{D}, e_i) \rightarrow (\mathcal{Q}, \mathcal{D}, e_j)$, where $e \in \mathcal{E}$ describes the system state (queue and device) which changes with respect to the associated event.

4.1 Abstraction Level

In the present section the job and device concepts adopted in the simulation model is described. As previously mentioned, a \mathcal{H} -system is composed by a finite collection, \mathcal{D} , of heterogeneous compute units used in the simulation. The collection is a tuple, defined as: $\mathcal{D} = (d_i, \dots, d_{ncpus}, d_j, \dots, d_{ngpu})$, where *ncpus* and *ngpus* represent the number of CPUs and GPUs respectively. Each device is qualified by a set of attributes:

- Id*: unique device numeric identifier
- Type*: CPU or GPU
- Status*: describes the device's status (busy or idle)
- Job*: identifies the running Job
- Cpulock*: provides the GPU Id when a CPU is coordinating a Job running on GPU, Job is running on such device. It is set to 0 when the device is not coordinating the GPU Job execution
- Statistics*: stores some device statistical information (such as the total occupation time and the number of processed jobs)

Each attribute value may change according to a device operation. The most significant operations are:

- Run Job*: a Job is coupled with a free device. To run a GPU job one CPU device must be free to control its execution.
- Terminate Job*: the Job is released and the device status switches over to free status.
- Status*: get the device status. This operation return the Job Id if the device is busy.

For the simulation model, a Job is a collection of **temporal** and **type attributes** that define its type (such as RealTime or CPU Bound). In general, the temporal attributes describe the Job evolution during the simulation. The temporal attributes are:

- Ta : indicates the arrival time of the Job into the system. In Section 4.3 is discussed how to select the input value according to the probability distribution.
- Tq : indicates the arrival time of the Job in the queue.
- Te : indicates the instant in which a queued Job starts the execution for the first time. Such value allows to compute the mean of the job's waiting time in the queue.
- Tr_{cpu} ,
 Tr_{gpu} : indicate, respectively, the service time in CPU and in GPU devices. Such values, unknown by the scheduler, represent the time required by the devices to satisfy the Job request. The values assigned to such attributes are discussed in section 4.3; however, the range of assigned values depends on the type attributes.
- Tf : indicates the exit time of the job from the system.

In addition to temporal attributes, the type attributes characterize the Jobs and the simulation process. Such attributes affect the service time and the type of device able to execute it. The types of Job defined are: *Realtime* and *User*. The RealTime Job represents the generalization of traditional Operating System RealTime job. This type of job is characterized by a low service time associated with an high execution priority. In our model it is assumed that the RealTime jobs are executed exclusively on a CPU device (more details are given in Section 4.2).

The second type of Job (User Job) generalizes the CPU Bound Job treated in Section 3. Generally, the User Job is executed by traditional devices (CPUs) or GPU devices with different service time depending on the application speedup.

According to the tuning input system, some User Job can be classified as GPU User Job with Tr_{gpu} service time or as CPU User Job with Tr_{cpu} service time. Furthermore, the GPU User Job can be run into CPU device, in this case the service time will be determined by the Tr_{cpu} value. Such assumption is based on the fact that the Job is a real OpenCL application and consequently it represents a runnable application on heterogeneous devices. In the simulation model, as well as in a real system, to execute this kind of Job both a CPU device (devoted to the Host Program execution) and a GPU (devoted to the Kernel Program execution) must be available. In the simulation process the Tr_{gpu} value takes into account both the Host Program execution time and the Kernel Program execution time. In Table 1 a summary of the Job types associated to the simulation model are shown.

Table 1. Resume of the Job types

Type	Device	Classification	Time of service
Realtime	CPU	-	Tr_{cpu}
User CPU	CPU or GPU	CPU	Tr_{cpu} or Tr_{gpu} with $Tr_{cpu} < Tr_{gpu}$
User GPU	GPU or CPU	GPU	Tr_{gpu} or Tr_{cpu} with $Tr_{gpu} < Tr_{cpu}$

As mentioned, in our model the Job classification is predetermined and we suppose by default the real system knows which jobs can be executed in GPU or CPU devices. The assignment of a class to a Job, changes the scheduler behavior and influences the system performances. To evaluate the performance of a system which is not aware of the job-device mapping, a system parameter (*ClassificationErrorRate*) is added. Such parameter introduces a degree of misclassification [14] of Jobs into the system, according to an input tuned probability (classification performance). A classification error can affect both a CPU and GPU User Job:

- Classification error of GPU User Job:* the classifier marks the GPU User Job as CPU with the service time $Tr_{gpu} < Tr_{cpu}$.
- Classification error of CPU User Job:* the classifier marks the CPU User Job as GPU with the service time $Tr_{gpu} > Tr_{cpu}$.

In our case, the scheduler, described in Section 4.2, is not able to detect the classification errors, since it ignores the Job service time.

Finally, a job is defined by **priority attributes**. The absolute Job priority depends on a combination of such attributes. Once properly tuned, it allows the simulation of different priority assignment strategies of classical scheduling algorithms such as CFS or O(1) [4,15]. In particular, the attribute *Job Priority (jp)*, indicates the static priority of the job during the entire life cycle, while *Dynamic Priority (dp)*, varies during the Job life cycle.

4.2 Scheduling Disciplines Implemented

In this section a simple scheduling discipline for the heterogeneous system, implemented in the simulator as the first use case, is shown. The policy adopted arrange the Job in a single no-preemptive priority queue. Generally, the selected Job to run can use the device for a quantum-time (or a credit), *qt*, before being released, as in a classical time-shared system [16]. The value of *qt* depends on the quantum-time policy described below:

<i>Static or Dynamic</i>	In static fashion the qt , once set, will not change during scheduling process. Otherwise (dynamic) the qt value may change, within a range, according to some criteria (for example considering the number of jobs in the queue).
<i>Homogeneous or Heterogeneous</i>	In homogeneous fashion, the qt value is the same for each Job, while in the heterogeneous one, each job has a different qt value called credit.
<i>Miscellaneous</i>	a combination of the two approaches above mentioned.

Our strategy enables the use of homogeneous static or dynamic qt values. The strategies of scheduling are summarized in the following rules:

Rule 1 *Each Realtime Job has higher priority than User job.*

Rule 2 *Each Realtime Job runs on a CPU device.*

Rule 3 *Each GPU User Job runs on a GPU device if it is free, otherwise on a CPU device.*

Rule 4 *Each CPU User Job runs on a CPU device.*

Rule 5 *Each Job executed on a CPU device can spend maximum qt cputime before being released.*

The next job to run is selected according to the following sequence:

1. The GPU User Job with higher priority is extracted from the queue and it is executed by the first available GPU device. The Rule 3 is ensured.
2. The Job with higher priority is extracted if all GPU devices are busy.
3. If the Job with higher priority is a GPU User Job and all GPU devices are busy, it is executed by CPU device, according to Rule 3.
4. If the Job with higher priority is a CPU User Job, it is executed according to Rule 4.

Finally, a CPU job execution is released and the job is re-queued if qt CPU time is spent and at the same time the Job priority is decreased. To avoid the starvation problem [17] the priority of enqueued jobs is incremented. However, to ensure Rule 1 the system enforces to Realtime jobs a higher priority than user job during the updating priority phase. The job with same priority runs on the device in a round-robin fashion.

4.3 Input Parameters

The simulation process depends on user-specified input parameters. The user can specify the following parameters:

n_{cpus}, n_{gpus} :	number of CPUs and GPUs devices (to simulate the hardware of the system).
Ta parameters:	mean inter-arrival time of Jobs into the system.
$T_{service}$ parameters:	mean and standard deviation of jobs service time for RealTime and User jobs. The system load is directly proportional to $T_{service}$ and Ta values.
Realtime rate:	rate of Realtime jobs relative to the total according to probability threshold.
speedup:	min and max speedup for GPU execution respect to CPU execution. According to [18] the range is between 9 and 130 depending on the type of application.
class err:	classification error rate, mean the rate of job correctly classified by the system. When the classifier does not make mistakes (100% success rate), it is assumed to simulate a system that knows which applications run in GPU fashion.

Other parameters such as quantum time and priority parameters are needed and depend on scheduling policy. The system jobs are generated on the basis of the previous parameter during the initialization phase.

4.4 Event and Work-Flow

The Simulator design is based on events which characterize the Job scheduling in a queue. The proposed simulation model handles the following events:

<i>Enqueue</i> :	define the arrival (submission) of the Job into the system.
<i>Run</i> :	after the submission and the activation of the scheduler is triggered extracting one or several Jobs from the queue to assign them to eventual available devices.
<i>Reschedule</i> :	define the activation of the scheduler having the resources busy to update the priority of the waiting jobs.
<i>Requeue</i> :	occurs when a Job has spent the qt CPU time it is released and its priority is decreased.
<i>Finalize</i> :	identify the exit of the Job from the system if its work is completed.

Each event is characterized by the time (t_{event}) which identifies the instant in which the event occurs. Every time an event is created it is inserted in a queue on the basis of its t_{event} value. The system extracts from the queue the event having the lowest t_{event} and executes the actions associated to the event type.

The simulation work-flow consists of two main phases. During the first step, at the same time of the initialization of the attributes of the job, it is created an Enqueue event type associated with the job and an event time corresponding to the value of Ta of the Job. The initialization ends when all jobs have been initialized and the corresponding Enqueue events were included in the event queue. During the second phase the events are consumed and produced depending on

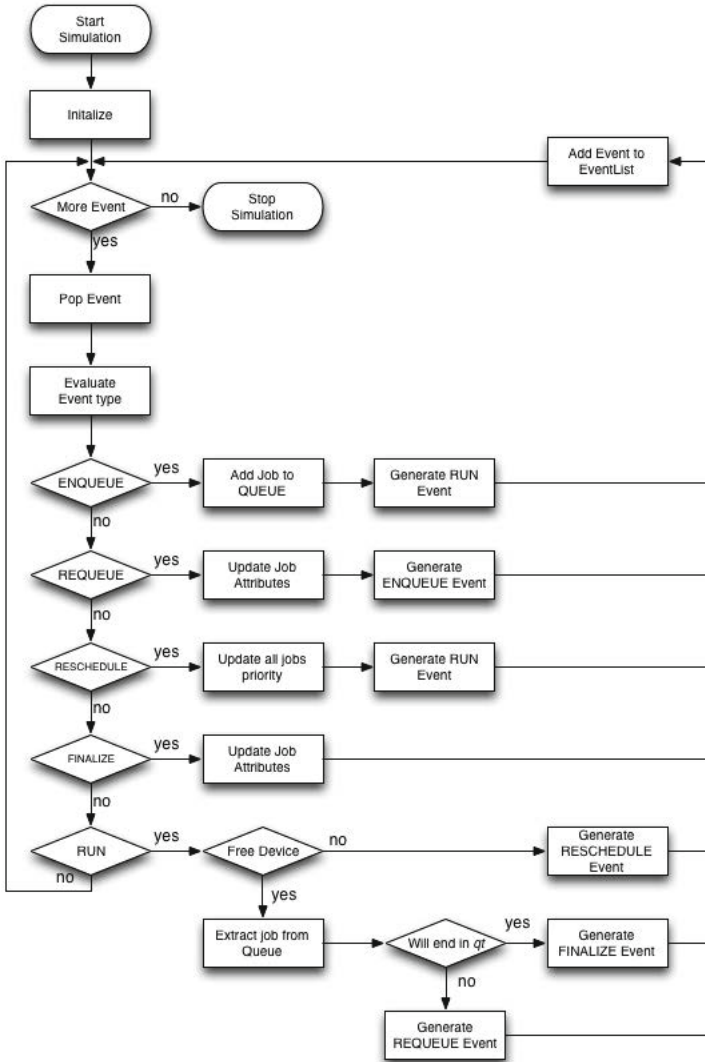


Fig. 1. Simulator work-flow

the characteristics of the job scheduling and the policy adopted. The simulation ends when all events have been consumed or when the simulation time exceeds a given threshold. In Figure 1 it is shown the operation flow of the simulation phase for each type of event extracted from the event queue.

When an event of Enqueue type occurs, the job associated to the event is inserted into the priority queue of the system and an event of type Run is generated with event time equal to $T_{CurrentEvent} + t_{sched}$ with t_{sched} representing the activation time of the scheduler. The event type Run, which is not associated to any Job, activates the scheduler, and considering the system status (described

considering both the queue and the state of the device) performs the following actions:

- Queue is not empty and resources busy:* if there are not available devices to execute the Job, an event of type Reschedule is generated with event time equal to $t_{event} + t_{sched}$.
- Queue empty:* when there are not jobs in the queue, regardless the state of resources, it is not generated and nor inserted any new event.
- Queue is not empty and resources available:* a Job is extracted according to the Job scheduling policy (see [4.2](#)) and assigned to a device by placing it in a busy state; in case of the first execution it is initialized the T_e of the Job equal to the value of the event time. In general, if the service time of Job has a value lower than qt , or whether the job is assigned to a GPU device, an event of type Finalize, having a t_{event} equal to the service time of the Job itself, is created. Otherwise, it is generated an event of type Requeue with associated the Job and the time event equal to qt .

As mentioned, the occurrence of the Run event can lead to the generation of Reschedule, or Finalize Requeue events. The Reschedule event implies the evaluation of the priorities of all Jobs in the queue and the generation of an event of type Run. The strategy and the update of the priorities values are specified by the user in order to study the impact on the performance of the simulated system. The event Requeue and the event Finalize are tied with a job. As per the Requeue event, the following operations are performed:

1. Releasing of the running Job as it has exhausted its time quantum.
2. Decrease its priority according to the chosen strategy, since it has already consumed a resource.
3. Decrease the service time required for the completion of the Job.
4. Update statistical information.
5. Create an event of type Enqueue associated with the Job (resubmission).

If the Job has a service time less than the quantum time of the system, or it has been executed in GPU devices (assuming so that the suspension is not possible on that device), an event Finalize is generated. This event produces the following operations:

1. Releasing of the Job, which in this case has terminated its execution with the exit time of the system (T_f) equal to the sum of the execution time and the service time.
2. Storing the attributes values of the Job for the generation of statistics to evaluate the system performance.

From a single Job point of view, Figure 2 shows the transition of the events in the system. During the initialization, an event Enqueue is generated in a time equal to Ta . The submission to the queue leads to the generation of a Run event and the execution of the Job in a device with a time Te . If the Job service time is less than qt determined by the Scheduling policy, a Finalize event is generated in time $Te + Tservice$, which will represents the exit time from the system. If the service time of Job exceeds qt in such case an event of type Requeue is generated in time $Te + qt$ and an Enqueue event and the job will be resubmitted to the system until its completion.

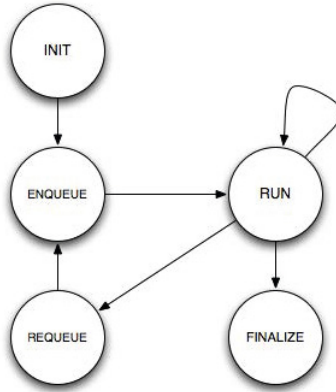


Fig. 2. Transition event

It is important to evaluate how the scheduling strategy influences the generation of system events. For instance to implement a strategy which immediately perform a Job once arrived in the system with the highest priority (preemptive strategy) it is necessary to delete the events generated by the Run event previously generated by the Job in execution. Consider for example that an event of type Run, led to the execution of the Job, has generated a Finalize event; the event should be eliminated from the event queue and replaced with a Requeue event with a time equal to the arrival time of Job with the highest priority that in turn would result in the generation of a Requeue or Finalize event.

5 Conclusions and Future Works

This work aims to study the impact of non-conventional devices as general purpose compute units (such as GPUs) in the scheduling process of the modern operating systems to improve the system performance. To achieve this goal a simulation framework to model events and to evaluate the scheduling policy for heterogeneous systems was presented, along with its implementation.

The simulator provides the following features:

1. Creation of the user-specified hardware in terms of number of CPUs and GPUs.
2. Generation of the system load, setting the number of jobs.
3. Tuning of the inter-arrival Job time.
4. Selection of the Job composition. It allows to specify the probability to generate a given number of Realtime, GPU User and CPU User Job.
5. Setting Classifier simulation.
6. Selection of *qt* strategy.

In the present version of the Simulator the scheduling policy has to be implemented and customized by the developer. A scheduling policy has been implemented as a use case. The Job are arranged in a single non-preemptive priority queue. The scheduling discipline is summarized in rules 1-5, described in Section [4.2](#).

Future works will focus on three main aspects: the first one will concern the study of inter-arrival of real system and the implementation of the linux scheduler (CFS) to validate the simulator. The second one will concern the addition of new improvements to the simulator such as a graphics interface and an automatic tools for the generation of charts for the analysis of the performance of the scheduling strategies. Finally other scheduling policies will be compared and new strategies will be defined and evaluated.

References

1. NVidia: NVIDIA CUDA - Compute Unified Device Architecture: Programming Guide (2011), <http://developer.nvidia.com/nvidia-gpu-computing-documentation>
2. Khronos OpenCL Working Group, The OpenCL Specification, version 1.0.29 (2008), <http://khronos.org/registry/cl/specs/openc1-1.0.29.pdf>
3. Vella, F., Cefalá, R., Costantini, A., Gervasi, O., Tanci, C.: Gpu computing in egi environment using a cloud approach. In: 2011 International Conference on Computational Science and Its Applications, pp. 150–155. IEEE (2011)
4. Pabla, C.: Completely fair scheduler. Linux Journal 2009(184), 4 (2009)
5. Lin, C., Lai, C.: A scheduling algorithm for gpu-attached multicore hybrid systems. In: 2011 5th International Conference on New Trends in, Information Science and Service Science (NISS), vol. 1, pp. 26–31. IEEE (2011)
6. Guevara, M., Gregg, C., Hazelwood, K., Skadron, K.: Enabling task parallelism in the cuda scheduler. Work (2009)
7. Jiménez, V.J., Vilanova, L., Gelado, I., Gil, M., Fursin, G., Navarro, N.: Predictive Runtime Code Scheduling for Heterogeneous Architectures. In: Sez nec, A., Emer, J., O’Boyle, M., Martonosi, M., Ungerer, T. (eds.) HiPEAC 2009. LNCS, vol. 5409, pp. 19–33. Springer, Heidelberg (2009)
8. Phatanapherom, S., Uthayopas, P., Kachitvichyanukul, V.: Dynamic scheduling ii: fast simulation model for grid scheduling using hypersim. In: Proceedings of the 35th Conference on Winter Simulation: Driving Innovation. pp. 1494–1500. Winter Simulation Conference (2003)

9. Buyya, R., Murshed, M.: Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience* 14(13-15), 1175–1220 (2002)
10. Casanova, H.: Simgrid: A toolkit for the simulation of application scheduling. In: *Proceedings of First IEEE/ACM International Symposium on Cluster Computing and the Grid 2001*, pp. 430–437. IEEE (2001)
11. Gupta, A., Tucker, A., Urushibara, S.: The impact of operating system scheduling policies and synchronization methods of performance of parallel applications. In: *ACM SIGMETRICS Performance Evaluation Review*, vol. 19, pp. 120–132. ACM (1991)
12. Law, A., Kelton, W.: *Simulation modeling and analysis*, vol. 3. McGraw-Hill, New York (2000)
13. Gere Jr., W.: Heuristics in job shop scheduling. *Management Science*, 167–190 (1966)
14. Japkowicz, N., Stephen, S.: The class imbalance problem: A systematic study. *Intelligent Data Analysis* 6(5), 429–449 (2002)
15. Brown, R.: Calendar queues: a fast 0 (1) priority queue implementation for the simulation event set problem. *Communications of the ACM* 31(10), 1220–1227 (1988)
16. Silberschatz, A., Galvin, P., Gagne, G.: *Operating system concepts*, vol. 4. Addison-Wesley (1998)
17. Tanenbaum, A., Tannenbaum, A.: *Modern operating systems*, vol. 2. Prentice Hall, New Jersey (1992)
18. Nickolls, J., Dally, W.: The gpu computing era. *IEEE Micro* 30(2), 56–69 (2010)

Influence of Topology on Mobility and Transmission Capacity of Human-Based DTNs

Danilo A. Moschetto², Douglas O. Freitas¹, Lourdes P.P. Poma¹,
Ricardo Aparecido Perez de Almeida¹, and Cesar A.C. Marcondes¹

¹ Computer Science Department – Federal University of São Carlos (UFSCar), Brazil

² Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - IFSP

moschetto@gmail.com,

{douglas_freitas, lourdes.poma, ricardoalmeida,
marcondes}@dc.ufscar.br

Abstract. Casual encounters among people have been studied as a means to deliver messages indirectly, using delay tolerant networks (DTNs). This work analyses the message forwarding in human-based DTNs, focusing on the topology of the mobility area. By using simulations, we evaluate the influence of the environment connectivity on the network performance. Several parameters have also been considered: network density, forwarding algorithm and storage capacity. In general, considering the already limited capacity of mobile devices and a reduced network density, the mobility environment interconnectivity seems to have a relevant effect in message delivery rates.

1 Introduction

The current Internet architecture is based primarily on the TCP/IP protocol stack and the end-to-end argument [1]. This means the communication intelligence is kept mostly at the source and destination hosts, while the network core is (relatively) simplified.

Even being one of the biggest factors for the success of the Internet, this simplicity is also a major factor for its ossification [2]. High-delay and intermittent link connection, observed by mobile hosts, are particularly difficult to handle with the established protocols. The desire to support newer envisioned communication models and to fix existing deficiencies of the current Internet have then stimulated new protocols, such as the creation of delay-tolerant networks (DTNs) [3]. Communications in DTNs may occur either directly, when a link from a node to the infrastructure exists long enough, or using an ad-hoc, circumstantial, link between two nodes for indirect delivery.

Several uses of DTNs have been proposed, such as those described in [4], [5] and [6]. In general, DTNs are used in areas without a communication infrastructure (rural or remote), in areas with a damaged infrastructure (natural disasters or conflicts, for example) and even in urban areas, when the conventional network infrastructure is not accessible or does not provide the desired quality of service. Applications in different domains, such as games, tourist information and advertising can benefit from this form of communication.

The communication effectiveness in a DTN network is considered in respect with the delay and the probability of successful delivery of a message. Given the non-deterministic conditions under which communications in DTNs occur, different heuristics can be applied for message routing. Data about previous encounters between nodes, the transmission capacity per node, and power consumption and availability are commonly used in the forwarding decisions. In addition, other features can also be considered, such as buffer size, available wireless communication technologies, computing power, routing algorithm, node proximity, competition for the wireless channel, and obstacles to signal propagation.

In this paper, we consider the specific case of human-based DTNs in a central metropolitan region. In this type of network, non-technical factors, such as user mobility pattern, group formation, encounters and the existence of points of interest, can play a relevant role in the heuristics, which makes message forwarding challenging.

Although many of the relevant factors for message delivery have been previously evaluated individually [4], [7], [8], little attention has been devoted to the influence of the city area (the topology) of the cities on the efficiency of communication. This paper investigates the impact of the topology of cities in the transmission capacity of human-based DTNs. The obtained results show the relevance of this factor in the probability of successful delivery of messages and the average incurred delay.

An evaluation on the impact of the topology complexity of maps is presented, showing that changes in the connectivity of the streets influence the delivery rate. The studies were conducted using simulation and statistical analysis, varying parameters such as network density, buffer size, transmission technology, routing algorithm and the topology of the user mobility area.

This paper is organized as follows. In section 2 we present a discussion of routing messages in DTNs and related work. In Section 3 we show the simulation environment used. Next, in Section 4 we discuss mobility models and present the model used in this work. In Section 5 we perform a categorization of topologies according to characteristics of cities. We evaluate the influence of parameters (buffer, node density and routing protocols) in the transmission capacity of a DTN in Section 6. The influence of changes in the topology map is investigated in Section 7. Finally, in section 8 we conclude our work and present a discussion about future work.

2 Message Forwarding in DTNs

Message forwarding in human-based DTNs, when there is no communication infrastructure available, is based on transmission opportunities fostered when mobile devices are near. Different wireless transmission technologies can be used for data transmission, and the decision on when and to whom messages be forwarded may be based on different policies and criteria. The efficiency of a routing strategy can then be measured according to more or less resource consumption, such as temporary storage space and battery power. Minimizing resource consumption and maximizing delivery are desirable.

This work investigates the delivery capacity of a human-based DTN formed in urban environments, and analyzes the impact of the topology (interconnection of streets, avenues, landforms, etc.) of different cities in the communications. Given the

difficulties in parameterizing the data about the streets and the movement of people on the existing paths of a city, we chose not to use analytical models but to rely on simulations of the aspects of interest. Also, no real experiment with the intended proportions would be feasible.

Isolated characteristics of DTNs have been investigated and modeled in previous works. In [7], a DTN network formed by 3 nodes is modeled using queuing networks to evaluate the message delivery delay between nodes. The obtained results show that the delay is influenced by the contact time just in part of the transmission period. However, that study does not consider the topology of the movement area and is limited to a network with only three nodes, as the number of states for the queuing network grows exponentially.

The work in [9] presents a framework for the analysis of transmission capacity in mobile ad hoc networks, showing that if the node mobility process is stationary and ergodic, the movement pattern is sufficient to determine the transmission network capacity. That result is related to the hypothesis brought about by this work, once that if the movement is sufficient to determine the transmission capacity of a network, then clearly the restriction of movement due to the topological conditions must also be relevant.

With respect to works considering the topology of movement areas, [10] proposes a georouting protocol that implements a best effort geocast service based on GPS information and mobility maps. The protocol is evaluated by simulation and shows a slight improvement in delay and message delivery. The proposed heuristic was not tested.

The use of context information about the network and the environment is considered in [11]. The authors propose an approach to build resource maps to estimate the availability of resources (battery level, buffer space and bandwidth) of neighbor nodes in a DTN. Using their approach, network nodes can more effectively determine which neighbor nodes should receive messages, increasing the number of successful deliveries and reducing energy consumption and transmission delays. Another study, [8], uses simulation to evaluate the impact of different routing protocols for DTNs. The results suggest that using configurable context-based protocols increases the delivery rate. It seems that both studies, [11] and [8], could be extended to consider the topology of the circulation area in the routing decisions.

That observation justifies our effort to investigate the hypothesis presented here.

3 Simulation Environment

Given the wide range of simulation environments supported, the *The One* simulator [12], developed in Java, has been widely adopted in the study of DTN and was used in our evaluations. Among its main advantages considered, we may highlight the variety of routing modules supported, its extensibility, and the interoperability with different file formats, such as WKT files commonly used to represent the topology of cities.

A few extensions had to be developed and incorporated to the simulator to record the location of where each message delivery occurs during simulation. This was achieved by adding resources to use the coordinates provided by the class `DTNHost`, and the creation of a model, named *Stationary Movement*, to create reports with the

saved coordinates of message delivery. From the results produced by the modifications, it is possible to create a configuration script for the simulator so it may use the graphical display to show the points of delivery on the map used in the simulation.

Using WKT files, containing vector maps of cities using linestrings, we have been able to use the simulator to evaluate a variety of maps of metropolitan areas of major cities around the world. This has allowed us to experiment with different topologies and to consider their impacts on message delivery. The maps used were obtained from [13], and are based on the Open Street Map project.

The maps used in the tests correspond to an area of approximately 6 square km, and vary according to the number of streets and their interconnections. The area represented in the maps considers the central areas of the selected cities, comprising a microcosm of the urban area as a whole. Moreover, as this paper focus on the movement of people, longer distances, corresponding to the extended metropolitan areas of the cities, would not be sensible for walking are not relevant in this context.

It is not in the interest of this study to perform an exhaustive analysis of the specific routing algorithms, such as the Epidemic [14], which are used in this work. Information about this type of study can be found in [8]. A preliminary study of parameters sensitivity, such as simulation time and buffer size, required to weaken the influence of these in the used routing algorithms was also conducted. Aspects of energy consumption of the devices were not considered in our tests.

4 Mobility Models

Node mobility is one of the most studied parameters in human-based DTNs. In [15], it is argued that the Levy Walk mathematical model can be used to reproduce nuances of human movement, once humans, as also notices with other animals (zebras, for example), tend to have a mobility pattern based on short movements around points of interest, interleaved with eventual long movements to other distant points of interest.

The original Levy Walk model considers the random movement in an obstacle-free environment. This, however, differs from the movement of people in cities, which walk according to existing streets and avenues. Among the mobility models based on maps available in the The One simulator, the Shortest Path Map-based Movement (SPMBM) [12] was considered to have a pattern similar to Levy Walk, once it uses a selection method based on Dijkstra's Shortest Path Algorithm. Therefore, choosing the SPMBM for the experiments was considered reasonable, especially regarding the human behavior of frequently choosing the shortest path to a destination, according to prior knowledge about the city they live in.

In order to validate the correspondence between SPMBM and Levy Walk, preliminary simulations were performed comparing the results produced by the two models. For these simulations, we have considered a network with density as described in Sections 6 and 7, and an area of 6.25 square km, also similar to the area of all maps used in this work. To test the SPMBM algorithm, we have selected the map of Venice, which has a large number of streets and interconnections, contributing to randomness in the movement of people. For tests with Levy Walk, we considered the movement as happening in an open area. The results are presented in Table 1.

Table 1. Comparison between Levy Walk and Shortest Path models

	<i>Levy Walk</i>		<i>Shortest Path (Venice)</i>	
	Average	Standard Deviation	Average	Standard Deviation
Delivery Probability	83%	1%	88%	0,7%
Messages Transmitted	813719	1778	879501	1860
Number of encounters	6147	46	15032	421

The simulation time was approximately 4.5 h, which was enough time for the system to reach a stationary state. The network was formed by 600 nodes (people) that transmitted messages of 1 KB each other via Bluetooth technology, and using the Epidemic algorithm for message delivery. Epidemic routing presents high delivery capacity, obtained by replications of each message. The buffer capacity of each node was set to 1 GB, used for intermediary storage of in-transit messages. This amount of memory was considered to be available in most current devices. Each node can move to a distance of up to 1100 meters from its start point. The simulations were performed five times for each mobility model, using different seeds to generate random node positions.

Table 1 shows that the average delivery probability for both mobility models was similar, varying by only 5% and with little deviation. In addition, the number of effectively delivered messages remained close in both results (varying up to 8%). We have, however, found a significant difference when comparing the number of encounters, as the simulations using the SPMBM movement model was 2,4 times greater than the value obtained with Levy Walk. This result can be justified by the absence of physical barriers to restrict the node movement in simulations performed with Levy Walk. Once a greater number of movement possibilities are available, nodes tend to meet each other occasionally in Levy Walk model, differently than the SPMBM model, in which nodes were limited to just move through the existing streets of Venice.

Considering the results presented above, we realized that SPMBM model has a complementary behavior to Levy Walk, limiting the movement to only the streets, while maintaining the same delivery probability. Based on it, the SPMBM was chosen in the simulations.

5 Topology Categorization According to Its Complexity

In order to investigate if different topology complexities influence the likelihood of message delivery DTN networks, we performed a preliminary categorization that considers aspects of complexity of the topology graph of the cities.

Initially, 10 maps were randomly selected. Simulations were conducted with the maps of Cairo, Chennai, Karlsruhe (2 maps), Los Angeles, New Delhi (2 maps), Richmond, Tokyo and Venice. The simulations were performed using the following parameters: Epidemic routing, 600 nodes, 1KB message sizes, and transmissions using Bluetooth. Each experiment was repeated 5 times and focused on the determination of the number of encounters between nodes (persons). The results are shown in Figure 1.

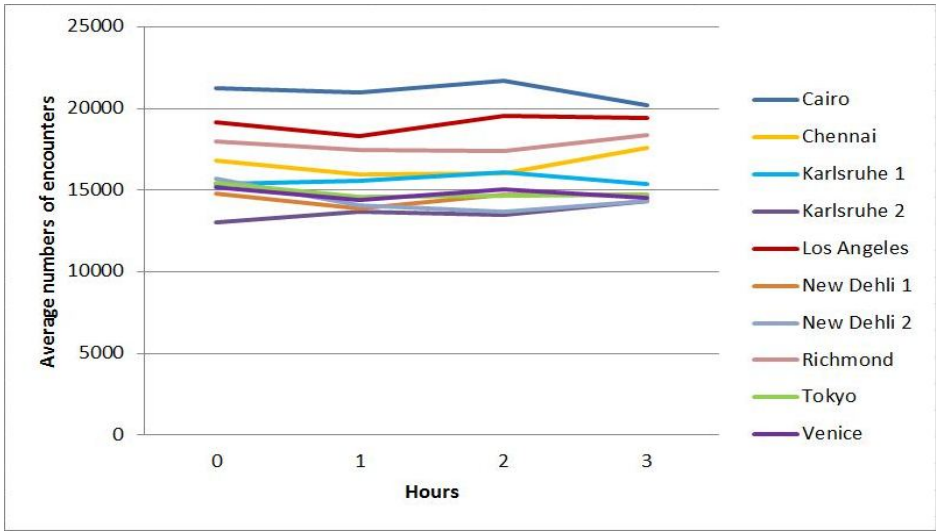


Fig. 1. Average number of encounters for 3-hour simulation

Figure 1 shows that the cities of Cairo and Los Angeles produced larger number of encounters in the experiments. The maps of Karlsruhe 2 and New Dehli 2 lead to fewer encounters. However, we see that these maps didn't show similar behavior during the simulations, which led to the choice of Venice and Tokyo as your representatives containing a smaller number of encounters. The maps of the selected cities are shown in Figure 2.

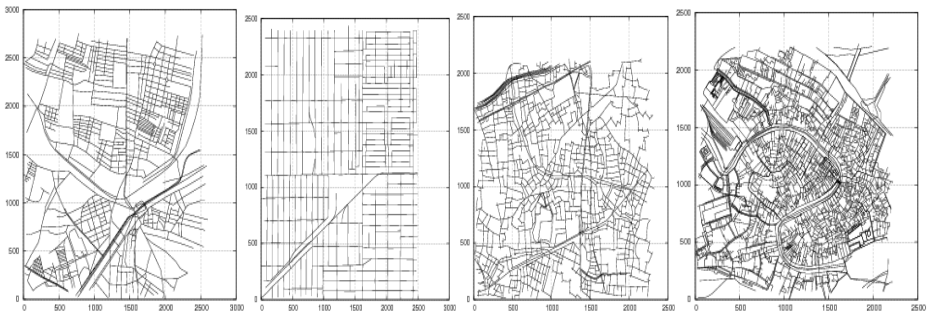


Fig. 2. Partial maps of Cairo, Los Angeles, Tokyo and Venice [13]

As can be seen from Figure 2, the maps of the cities of Cairo and Los Angeles appear to have a smaller number of streets and interconnections, compared to the maps of Tokyo and Venice. This impression was confirmed by comparing the graphs of street interconnections for each map, as we evaluated the total number of interconnections (points of a linestring on the map, which generally corresponds to intersections), and the connectivity degree. Table 2 and the graph in Figure 3 show, respectively, the number of interconnections of each map and its distribution according to the connectivity degree.

Table 2. Number of interconnections that compose each map

	Cairo	Los Angeles	Tokyo	Venice
Number of interconnections	1464	573	3655	7982

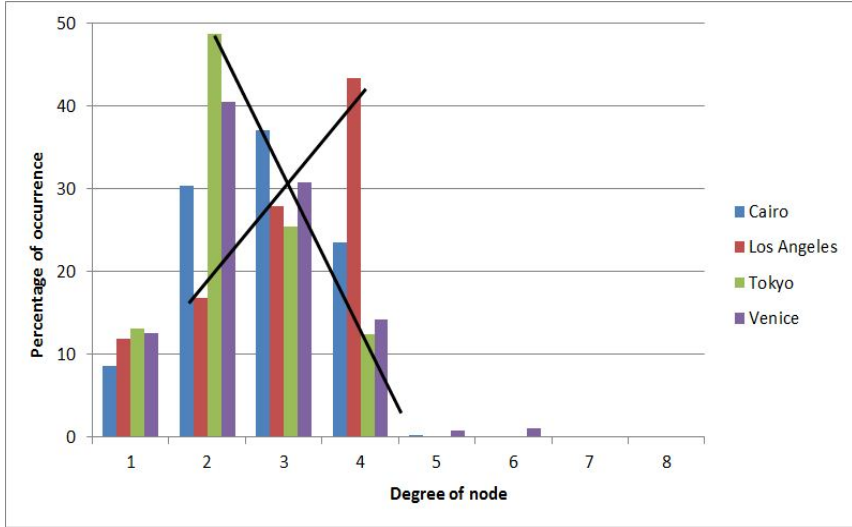


Fig. 3. Distribution according to the connectivity degree

Table 2 shows that the cities of Cairo and Los Angeles have a small number of interconnections than Tokyo and Venice, which could provide fewer opportunities for changing routes while walking. Additionally, Figure 3 shows that the city of Los Angeles presents an upward trend of connectivity, although with a smaller number of interconnections. In other words, there are more interconnections with high degree than with fewer. Tokyo has a downward trend, with a high number of interconnections with lower grade. Only Tokyo and Venice present interconnection degrees greater than 4, while Venice is the only city with grades 7 and 8 nodes.

The observed data suggests we could estimate the encounter rate of different cities based on their interconnection maps. Moreover, one could assume that, based on the number of encounters, that Tokyo and Venice should have lesser odds of delivery than Cairo and Los Angeles, regardless of other parameters considered in the data forwarding policy. The following Section presents an evaluation of this hypothesis.

6 Analysis of the Transmission Capacity in Human-Based DTNs

Having selected maps to represent cities of different interconnection degrees for use in our study, we have analyzed the influence of other parameters on message delivery. The results were obtained via simulation.

Each simulation was repeated five times, considering the transmission of messages ranging from 500KB to 1MB via Bluetooth. Each simulation corresponded to a period of 4.5 hours. The experimented variables consisted of: size of local buffer at the nodes, node density in relation to areas of the maps, and routing algorithm. The results obtained are summarized in the following subsections.

6.1 Influence of Buffer Size

The size of the local buffer at each node can influence the probability of delivery in a DTN, as small spaces can lead to more messages to be discarded without being forwarded. For instance, in order to increase the likelihood of delivery, the Epidemic algorithm generates multiple copies of a message to be transmitted, requiring, therefore, more space to prevent other outstanding message from being discarded.

We have evaluated the influence of buffer size when using the Epidemic routing algorithm. The tests were performed for the city of Cairo, which presented higher number of meetings and, therefore, greater chances of completing the transmissions. The results from the simulations are presented in Table 3.

Table 3. Results from varying the buffer size in a network with 600 nodes

Delivery Probability										
<i>Buffer(MB)</i>	2	4	8	16	32	64	128	256	512	1024
Average	5.91%	10.25%	21.15%	28.93%	40.46%	51.30%	65.02%	75.54%	86.17%	86.17%
Standard Deviation	0.99%	0.88%	1.03%	0.47%	2.17%	1.57%	5.06%	1.55%	0.74%	0.74%
Average Delay in seconds										
<i>Buffer (MB)</i>	2	4	8	16	32	64	128	256	512	1024
Average	3889	3843	3713	3567	3418	3309	2846	2404	2283	2283
Standard Deviation	553.4	454.00	459.43	211.87	164.33	87.67	231.28	76.42	70.16	70.16

The results in Table 3 show that the delivery probability increases as the buffer size grows, but remains stable after reaching 512 MB. In our simulations, larger buffer sizes did not produce better delivery rates or reduced delays before delivery.

6.2 Influence of Node Density on the Map

Whereas the message delivery in a human-based DTN is subject to the encounters, we aimed to evaluate the influence of network density in each scenario considered in this study. Thus, simulations were performed considering densities of 200, 400 and 600 nodes, each featuring 1 GB local buffers, given the results in 6.1. The results obtained are shown in Figure 4.

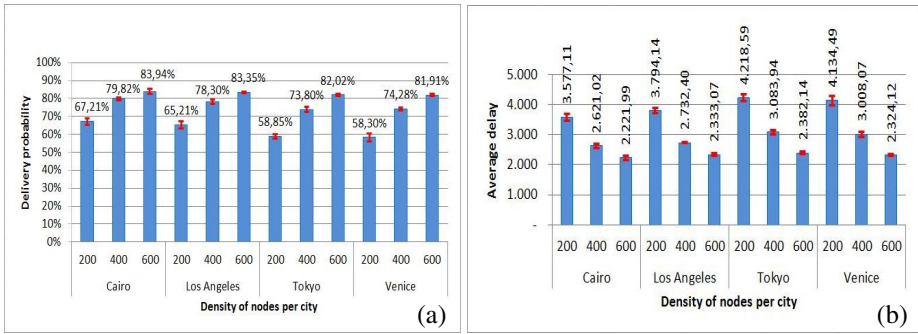


Fig. 4. Delivery Probability (a) and Average delay by node density (b)

The results in Figure 4 (a) show that even for small densities, the odds of delivery are higher than 50%. Also, one can note that, as the density increases, the probability of delivery also rises, reaching results up to 24 points higher.

Comparing the results obtained in each city, the city of Cairo had the best odds of delivery and the lowest delays for all densities evaluated, followed by Los Angeles, Tokyo and Venice. These results showed the influence of the topology of a city in the probability of successfully delivering a message. This can be seen as keeping the same settings for the network, and given a reasonable transmission time, Cairo showed the best results. On the other hand, as the density increases, the odds of delivery obtained and the average delay becomes very close for all cities examined, showing that the topology becomes less influential.

Continuing our analysis, Figure 5 shows the results of the number of encounters between network nodes for each density evaluated.

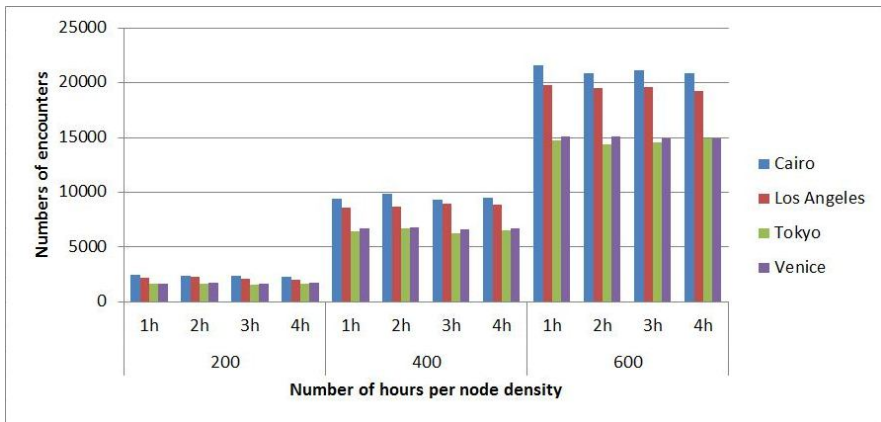


Fig. 5. Number of encounters per node density

As shown in Figure 5, the number of encounters between persons, and their devices carrying messages, increases as node density – crossings deriving from a corner –grows. For all densities evaluated, it is seen that the cities of Cairo and Los Angeles allow more encounters between the nodes, which contributes to a greater number of deliveries.

6.3 Influence of the Routing Protocol

In order to investigate whether the routing protocol used can mitigate the impact of the topology of a map in the number of transmissions and effective delivery, simulations were performed using the algorithms Epidemic and PROPHET [16]. The simulations consider DTNs formed by 200, 400 or 600 nodes, using the same values explained in previous Sections for the other relevant parameters.

The simulation results are presented in Figure 6, which shows that the Epidemic protocol provided better delivery odds than PROPHET for all studied scenarios. This differs from the results presented in [8], where it is shown that, for a network with limited resources (e.g. buffer size), the protocol PROPHET tends to provide better results. It is worth noting that in the context of this work, limitations as the buffer size had its impact mitigated.

Analyzing the odds of delivery obtained for the four cities of the study, we observed that both evaluated routing protocols had their best results in the cities of Cairo and Los Angeles, with variations that reach 10 percent (comparison of Cairo and Tokyo, with 200 nodes and PROPHET protocol).

Whereas the configuration parameters used were the same, we see that differences in the topology of the maps end up providing better or worse chances of delivery, although it is noticeable that, as node density increases, the influence of topology tends to be attenuated.

For further comparison, it is also possible to use the goodput (GP) metric, which measures the proportion of messages delivered to recipients in relation to the total volume of messages transmitted. For the tests, for example, in the city of Venice with 200 nodes, the protocol PROPHET obtained $gp = 1.02 * 10^{-2}$, while the protocol Epidemic obtained $gp = 0.84 * 10^{-2}$, which represents a variation larger than 20%.

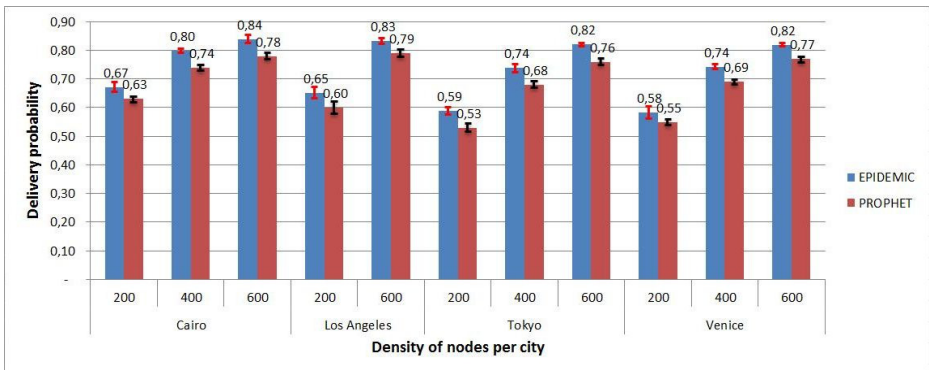


Fig. 6. Delivery Probability using Epidemic and PROPHET

7 Influence of Map Topology in Message Delivery

As observable from the previously presented results, effectiveness and delay vary for message delivery in the different cities analyzed. In order to investigate the influence of the topological restriction to urban mobility in human-based DTNs, we made hypothetical changes to the interconnections of one of these cities. Cairo was chosen due to the apparent reduced number of interconnections between its two parts, top and bottom, as shown in the map. By restricting the movement of people walking in the central area with few alternatives paths, the chances of encounters seem to grow, thereby generating an increased number of opportunities for message forwarding.

To evaluate this assumption, simulations were conducted considering the changes to the new map. These changes consisted in adding transition points between the top and bottom areas of the city shown in the map. The simulations used the same parameters defined in the previous sections. Figure 7 shows the hypothetical changes made to the map of Cairo map and the resulting connectivity degree of the interconnections.

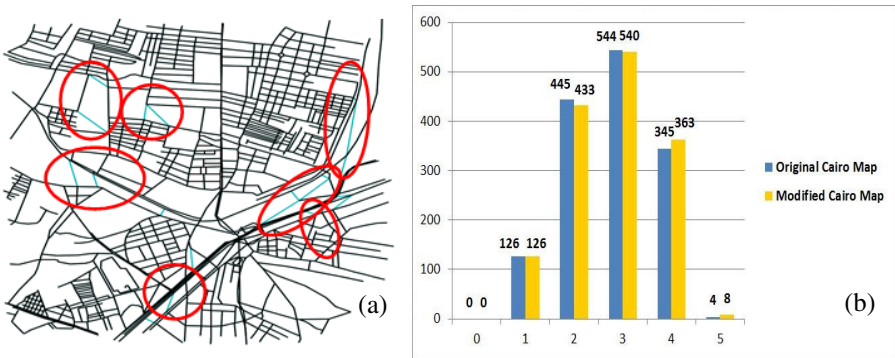


Fig. 7. Modified Cairo map (a) e node distribution according to the node degree (b)

In Figure 7 (a), it is possible to observe that a few interconnections were added to sections of the map. As shown in Figure 7 (b), the modifications made caused only few changes to the connectivity degree of the map.

The simulation results for this study are presented in Table 4.

Table 4. Simulations with the maps of Cairo and modified Cairo

	Cairo		Modified Cairo	
	Average	Standard Deviation	Average	Standard Deviation
Number of Encounters	21095	902	16970	994
Delivery Probability	84%	1,4%	81%	1%

The results presented in Table 4 show that the number of encounters and the delivery probability have decreased by 19% and 3%, respectively, in the simulations with the modified map of Cairo. These results show that small changes in topology can influence the results obtained in a human-based DTN. In order to observe the use of the new paths created, Figure 8 shows the original map of Cairo and its modified version, highlighting the places where the encounters between persons occurred in the simulations.

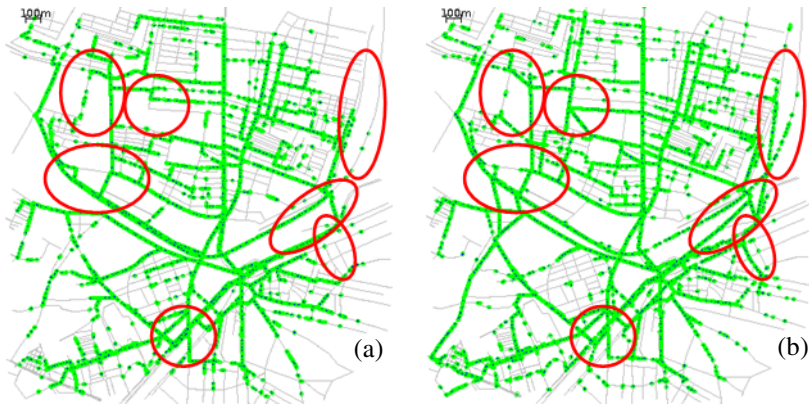


Fig. 8. Original (a) and modified (b) Cairo maps

Figure 8 (b) shows that the persons, carrying their mobile communication devices, have used the new paths included in the map. The green dots indicate the locations in which message forwarding happened. The difference in movement is caused by the creation of shorter routes between various points of interest in the map, which are used by the SPMBM algorithm. Those new path options influence the results obtained by the DTN.

The interconnection degrees of the maps, original and modified, did not change significantly. If changes had occurred to the periphery of the map, no significant changes would be expected in the efficiency of message delivery either.

8 Conclusion and Future Work

Human-based DTNs are a feasible alternative for routing messages in urban areas. Applications such as alerts, games and advertising can benefit from this form of message delivery. However, the efficiency of a network can be influenced by the restricted mobility generated by the topology of the walking area. The results from our analysis show that the interconnection of urban mobility elements (streets, avenues, etc.) is an important aspect to be considered, given the limited possibilities of movement. This limitation affects the probability of encounters between humans and may have consequences for any DTNs.

Considering the influence of the topology of the movement area, it was observed that for low-density DTNs (with up to 200 people), and all other network parameters being fixed, the topology is a more significant factor for message delivery. The connectivity degree of streets and preferential paths showed to be more relevant in this case. The scenarios evaluated in this study show that the delivery probability can vary up to 10% just with few changes in the topology.

On the other hand, if the network density (number of possible relays) is higher and all other operational conditions are maintained (buffer size, moving time), the topology does not present a relevant influence, being responsible for a maximum variation of 2% in the delivery probability.

For networks where the number of possible intermediate nodes is more limited, restricted to personal affinity, as can be observed in social networks, the delivery probability becomes more dependent on the topology. Delivery mechanisms that are more or less aggressive in terms of the replication of messages can be configured to use knowledge about of the connectivity degree of the topology. Similarly, power consumption can be reduced by avoiding unnecessary transmissions.

Adaptive routing algorithms based on an index composed by a list of traditional algorithms (Epidemic, PROPHET, Spread and Wait) may consider the structure of each city or microregion in their forwarding decisions. An example that applies this strategy could use online map services to extract the connectivity degree of a city and determine the appropriate algorithm. Another possibility is the dynamic change of parameters of traditional algorithms, also considering the connectivity of the movement area, which extends the results presented in [8].

References

1. Saltzer, J.H., Reed, D.P., Clark, D.D.: End-to-end arguments in system design. *ACM Transactions on Computer Systems (TOCS)* 2(4), 277–288 (1984), <http://dl.acm.org/citation.cfm?doi=357401.357402>
2. Spyropoulos, T., Fdida, S., Kirkpatrick, S.: Future Internet: fundamentals and measurement. In: *Proceedings of ACM SIGCOMM Computer Communication Review*, Tokyo, Japan, pp. 101–106 (2007), <http://doi.acm.org/10.1145/1232919.1232934>
3. Fall, K., Farrell, S.: DTN: an architectural retrospective. *IEEE Journal on Selected Areas in Communications* 26, 828–836 (2008), <http://dx.doi.org/10.1109/JSAC.2008.080609>
4. Juang, P., et al.: Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In: *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, San Jose, CA, USA, vol. 30, pp. 96–107 (2002), <http://dx.doi.org/10.1145/635506.605411>
5. Shah, R., Roy, S., Jain, S., Brunette, W.: Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks. In: *Proceedings of 1st IEEE International Workshop on Sensor Network Protocols and Applications*, Anchorage, AK, USA, pp. 30–41 (2003), <http://dx.doi.org/10.1109/SNPA.2003.1203354>

6. Burgess, J., Gallagher, B., Jensen, D., Levine, B.N.: MaxProp: Routing for vehicle-based disruption-tolerant networks. In: Proceedings of 25th IEEE INFOCOM, Barcelona, Spain, pp. 1–11 (2006), <http://dx.doi.org/10.1109/INFOCOM.2006.228>
7. Al Hanbali, A., de Haan, R., Boucherie, R.J., van Ommeren, J.-K.: A Tandem Queuing Model for Delay Analysis in Disconnected Ad Hoc Networks. In: Al-Begain, K., Heindl, A., Telek, M. (eds.) ASMTA 2008. LNCS, vol. 5055, pp. 189–205. Springer, Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-68982-9_14
8. Oliveira, E., Silva, E., Albuquerque, C.V.N.: Promovendo adaptação a contextos em DTNs. In: Proceedings of Simpósio Brasileiro de Redes de Computadores, Campo Grande, MS, Brazil (2011), http://sbrc2011.facom.ufms.br/files/main/ST03_1.pdf
9. Garetto, M., Giaccone, P., Leonardi, E.: On the Capacity Region of MANET: Scheduling and Routing Strategy. *IEEE Transactions on Vehicular Technology* 58, 1930–1941 (2009), <http://dx.doi.org/10.1109/TVT.2008.2004621>
10. Piorkowski, M.: Mobility-centric geocasting for mobile partitioned networks. In: Proceedings of IEEE International Conference on Network Protocols, Orlando, FL, USA, pp. 228–237 (2008), <http://dx.doi.org/10.1109/ICNP.2008.4697041>
11. Sandulescu, G., Schaffer, P., Nadjm-Tehrani, S.: Vicinity resource cartography for delay-tolerant networks: A holistic perspective. In: Proceedings of Wireless Days (WD), 2010 IFIP, Venice, Italy, pp. 1–7 (2010), <http://dx.doi.org/10.1109/WD.2010.5657725>
12. Keränen, A., Ott, J., Kärkkäinen, T.: The ONE Simulator for DTN Protocol Evaluation. In: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, Rome, Italy, pp. 55:1–55:10 (2009), <http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5674>
13. Mayer, C.P.: osm2wkt - OpenStreetMap to WKT Conversion (November 2010), <http://www.tm.kit.edu/~mayer/osm2wkt/>
14. Vahdat, A., Becker, D.: Epidemic routing for partially connected ad hoc networks. Technical Report CS-2000-06, Duke University (2000), <http://isg.cs.duke.edu/epidemic/epidemic.pdf>
15. Rhee, I., Shin, M., Hong, S., Lee, K., Kim, S.J., Chong, S.: On the Levy-walk Nature of Human Mobility. *IEEE/ACM Transactions on Networking*, 630–643 (2011), <http://dx.doi.org/10.1109/TNET.2011.2120618>
16. Lindgren, A., Doria, A., Schelén, O.: Probabilistic Routing in Intermittently Connected Networks. *ACM SIGMOBILE Mobile Computing and Communications Review* 7, 19–20 (2003), <http://dx.doi.org/10.1145/961268.961272>

Towards a Computer Assisted Approach for Migrating Legacy Systems to SOA

Gonzalo Salvatierra², Cristian Mateos^{1,2,3},
Marco Crasso^{1,2,3}, and Alejandro Zunino^{1,2,3}

¹ ISISTAN Research Institute

² UNICEN University

³ Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)

Abstract. Legacy system migration to Service-oriented Architectures (SOA) has been identified as the right path to the modernization of enterprise solutions needing agility to respond to changes and high levels of interoperability. However, one of the main challenges of migrating to SOA is finding an appropriate balance between migration effort and the quality of resulting service interfaces. This paper describes an approach to assist software analysts in the definition of produced services, which bases on the fact that poorly designed service interfaces may be due to bad design and implementation decisions present in the legacy system. Besides automatically detecting common design pitfalls, the approach suggests refactorings to correct them. Resulting services have been compared with those that resulted from migrating a real system by following two classic approaches.

Keywords: Services-oriented architecture, web services, legacy system migration, direct migration, indirect migration, semi-automatic cobol migration.

1 Introduction

From an operational standpoint, many enterprises and organizations rely on out-dated systems developed using old languages such as COBOL. These kind of systems are known as *legacy systems*. Still, enterprises have to face high costs for maintaining their legacy systems mainly because three factors [1]. First, these systems usually run on (expensive) mainframes that must be rented. Second, it is necessary to continuously hire and retain developers specialized in legacy technologies, which is both expensive and rather difficult. Third, in time these old systems have suffered modifications or upgrades for satisfying variable business goals. For example, most banks nowadays offer Home Banking services, though most bank systems were originally written in COBOL. Therefore, it is common to find a pre-Web, 50 year old technology working alongside modern platforms (e.g. JEE or .Net) within the same information system.

In this sense, migration becomes a necessity. Currently, the commonest target for migrating legacy systems is SOA (Service-Oriented Architecture) [2], by which systems are composed of pieces of reusable functionalities called *services*. Services are usually materialized through independent server-side “components” –i.e. Web Services [3]– that are exposed via ubiquitous Web protocols. Once built, Web Services can be remotely composed by heterogeneous client applications, called *consumers*.

Recent literature [4] identifies two broad approaches for migrating a legacy system: *direct migration* and *indirect migration*. The former consists of simply wrapping legacy system programs with Web Services. This practice is cheap and fast, but it does not succeed in completely replacing the legacy system by a new one. On the other hand, indirect migration proposes completely re-implementing a legacy system using a modern platform. This is intuitively expensive and time consuming because not only the system should be reimplemented and re-tested, but also the business logic should be reverse-engineered because system documentation could have been lost or not kept up-to-date.

In SOA terms, an important difference between direct migration and indirect migration is the quality of the *SOA frontier*, or the set of WSDL documents exposed to potential consumers after migration. Web Service Description Language (WSDL) is an XML standard for describing a service interface as a set of operations with input and output data-types. Although service interfaces quality is a very important factor for the success of a SOA system [5][6][7], most enterprises use direct migration because of its inherent low cost and shorter time-to-market, but derived WSDLs are often a mere low-quality, Web-enabled representation of the legacy system program interfaces. Then, services are not designed with SOA best design practices, which ensure more reusable interfaces. On the other hand, indirect migration provides the opportunity to introduce improvements into the legacy logic (e.g., unused parameters and duplicate code elimination) upon migrating the system, therefore improving the SOA frontier.

In this paper, we propose an approach called *assisted migration* that aims at semi-automatically obtaining SOA frontiers with similar quality levels to that of indirect migration but by reducing its costs. The approach takes as input a legacy system migrated using direct migration, which is a common scenario, and performs an analysis of possible refactoring actions to increase the SOA frontier quality. Although assisted migration does not remove the legacy system, the obtained SOA frontier can be used as a starting point for re-implementing it. This allows for a smoother system replacement because the (still legacy) implementation can be replaced with no harm to consumers since the defined service interfaces remain unmodified.

To evaluate our approach, we used two real SOA frontiers obtained by directly as well as indirectly migrating the same legacy COBOL system [1], which is owned by a large Argentinean government agency. We applied the assisted migration approach by feeding it with the direct migration version of the original system. After that, we compared the three obtained SOA frontiers in terms of cost, time, and quality. The results show that our approach produces a SOA frontier nearly as good as that the indirect migration, but at a cost similar to that of direct migration.

2 Automatic Detection of SOA Frontier Improvement Opportunities

We have explored the hypothesis that enhancing the SOA frontier of a migrated system can be done in a fast and cheap manner by automatically analyzing its legacy source code, and supplying software analysts with guidelines for manually refining the WSDL documents of the SOA frontier based on the bad smells present in the source code. This is because many of the design problems that occur in migrated service interfaces may be due to design and implementation problems of the legacy system.

The input of the proposed approach is a legacy system source code and the SOA frontier that result from its direct migration to SOA, concretely the CICS/COBOL files and associated WSDL documents. Then, this approach can be iteratively executed for generating a new SOA frontier at each iteration. The main idea is to iteratively improve the service interfaces by removing those *WSDL anti-patterns* present in them.

A WSDL anti-pattern is a recurrent practice that prevents Web Services from being discovered and understood by third-parties. In [8] the authors present a catalog of WSDL anti-patterns and define each of them way by including a description of the underlying problem, its solution, and an illustrative example. Counting the anti-patterns occurrences within a WSDL-based SOA frontier then gives a quantitative idea of frontier quality, because the fewer the occurrences are, the better the WSDL documents are in terms of reusability, thus removing WSDL anti-patterns root causes represents SOA frontier improvement opportunities. Note that this kind of assessment also represents a mean to compare various SOA frontiers obtained from the same system.

The proposed approach starts by automatically detecting potential WSDL anti-patterns root causes within the migrated system given as input (“Anti-patterns root causes detection” activity). Then, the second activity generates a course of actions to improve the services frontier based on the root causes found (“OO refactorings suggestion” activity). The third activity (“OO refactorings application”) takes place when software analysts apply all or some of the suggested refactoring actions. Accordingly, at each iteration a new SOA frontier is obtained, which feeds back the first activity to refine the anti-patterns root causes detection analysis. Figure 1 depicts the proposed approach.

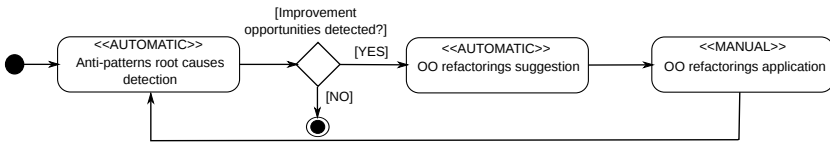


Fig. 1. Software-assisted SOA frontier definition activities

2.1 WSDL Anti-patterns Root Causes Detection

The anti-pattern root causes detection activity is performed automatically, since manually revising legacy system source code is a cumbersome endeavor. To do this, we have defined and implemented the ten heuristics summarized in Table 1. A defined heuristic receives the implementation or the WSDL document of a migrated transaction and outputs whether specific evidence of anti-patterns root causes is found.

We based on the work of [7] for designing most of the heuristics proposed in this paper, since [7] presents heuristics for automatically detecting the anti-patterns in a given WSDL document. Additionally, we have specially conceived heuristics 7 and 8 for analyzing COBOL source, while heuristics 6 and 9 for supporting two relevant anti-patterns that had not been identified in [7]. Concretely, the heuristic 6 analyzes names and data-types of service operations parameters to look for known relationships between names and types. Given a parameter, the heuristic splits the parameter name by basing on classic programmers’ naming conventions, such as Camel Casing and

Table 1. Heuristics for detecting evidence of WSDL anti-patterns root causes

Id and description	Input	True when
1. Look for comments in WSDL <documentation> elements	WSDL document	At least one operation lacks documentation.
2. Search inappropriate names for service elements	WSDL document	The length of a name token is lower than 3 characters, or when token refers to a technology, or when an operation name contains two or more verbs or an argument name contains a verb.
3. Detect operations that receive or return too many parameters	WSDL document	At least one operation input/output has more than P parameters.
4. Look for error information being exchanged as output data	WSDL document	An output message part has any of the tokens: "error", "errors", "fault", "faults", "fail", "fails", "exception", "exceptions", "overflow", "mistake", "misplay".
5. Look for redundant data-type definitions	WSDL document	At least two XSD data-types are syntactically identical.
6. Look for data-types with inconsistent names and types	WSDL document	The name of a parameter denotes a quantity but it is not associated with a numerical data-type (e.g. numberOfChildren:String).
7. Detect not used parameters	COBOL code	At least one parameter is not associated with a COBOL MOVE statement.
8. Look for shared dependencies among services implementations	COBOL code	The list of COBOL programs that are copied, or included, or called from two or more service implementations is not empty.
9. Look for data-types that subsumes other data-types	WSDL document	An XSD complex data-type contains another complex XSD data-type, or a list of parameters subsumes another list of parameters.
10. Detect semantically similar services and operations	WSDL document	A vectorial representation of the names and associated documentation of two services or operations, are near in a vector space model.

Hungarian notation. Each name token is compared to a list of keywords with which a data-type is commonly associated. For example, the token "birthday" is commonly associated with the XSD built-in `xsd:date` data-type, but the token "number" with the `xsd:int` data-type. Therefore, the heuristic in turn checks whether at least one name token is wrongly associated with the data-type.

The heuristic 7 receives the COBOL code of a migrated program and checks whether every parameter of the program output COMMAREA is associated with the COBOL MOVE assignment statement. In other words, given a COBOL program, the heuristic retrieves its output COMMAREA, then gets every parameter from within it (including parameters grouped by COBOL records), and finally looks for MOVE statements having the declared parameter. One temporal limitation of this heuristic is that the search for MOVE statements is only performed in the main COBOL program, whereas copied or included programs are left aside, and those parameters that are assigned by the execution of an SQL statement are ignored by this heuristic.

The heuristic 8 receives two COBOL programs as input. Then, for both programs it separately builds a list of external COBOL programs, copies, and includes, which are called (it looks for the CALL reserved word) from the main program, and finally checks whether the intersection of both lists is empty or not.

The heuristic 9 receives a WSDL document as input and detects the inclusion of one or more parameters of a service operation in the operations of another service. To do this, parameter names and data-types are compared. For comparing names classic text preprocessing techniques are applied, namely split combined words, remove stop-words, and reduce names to stems. For comparing data-types the heuristic employs the algorithm named *Redundant Data Model*, which is presented in [7].

Once we have all the evidence gathered by the heuristics, the mere presence of anti-pattern root causes represent opportunities to improve a SOA frontier. Formally, $id \rightarrow opportunity_i$ means that $opportunity_i$ may be present when the heuristic id detects its associated root causes (i.e. outputs 'true'):

- 4 \rightarrow *Improve error handling definitions*
- 8 \rightarrow *Expose shared programs as services*
- 10 \rightarrow *Improve service operations cohesion*

The first rule is for relating the opportunity to split an output message in two, one for the output data and another for fault data, when there is evidence of the output message conveying error data (heuristic 4). The second rule unveils the opportunity to expose shared programs as Web Services operations, when a COBOL program is called from many other programs, i.e. its fan-in is higher than a threshold T (heuristic 8). The third rule associates the opportunity to improve the cohesion of a service with evidence showing low cohesion among service operations (heuristic 10).

In the cases shown below, however, the evidence gathered by an heuristic does not support improvement opportunities by itself, and two heuristics must output 'true':

- 8 and 9 \rightarrow *Remove redundant operations*
- 1 and 2 \rightarrow *Improve names and comments*

The first rule is for detecting the opportunity to remove redundant operations, and is fired if two or more COBOL programs share the same dependencies (heuristic 8) but also the parameters of one program subsume the parameters of the other program (heuristic 9). The rationale behind this rule is that when two service operations not only call the same programs, but also expose the same data as output –irrespective of the amount of exposed data– there is an opportunity to abstract these operations in another unique operation. The second rule indicates that there is the opportunity to improve the descriptiveness of a service operation when it lacks documentation (heuristic 1) and its name or the names of its parameters are inappropriate (heuristic 2).

Finally, the rule *3 or 5 or 6 or 7 \rightarrow Improve business object definitions* combines more than one evidence for readability purposes, and it is concerned with detecting an opportunity to improve the design of the operation in/out data-types. The opportunity is suggested when the operation exchanges too many parameters (heuristic 3), there are repeated data-type definitions (heuristic 5), the type of a parameter is inconsistent with its name (heuristic 6), or there are unused parameters (heuristic 7).

2.2 Supplying Guidelines to Improve the SOA Frontier

The second activity of the proposed approach consists of providing practical guidelines to apply detected improvement possibilities. These guidelines consist of a sequence of

steps that should be revised and potentially applied by software analysts. The proposed guidelines are not meant to be automatic, since there is not a unique approach to build or modify a WSDL document and, in turn, a SOA frontier [9].

The cornerstone of the proposed guidelines is that classic Object-Oriented (OO) refactorings can be employed to remove anti-patterns root causes from a SOA frontier. This stems from the fact that services are described as OO interfaces exchanging messages, whereas operation data-types are described using XSD, which provides some operators for expressing encapsulation and inheritance. Then, we have organized a sub-set of Fowler et al.'s catalog of OO refactorings [10], to provide a sequence of refactorings that should be performed for removing each anti-pattern root cause.

Table 2. Association between SOA frontier refactorings and Fowler et al.'s refactorings

SOA Frontier Refactoring	Object-Oriented Refactoring
Remove redundant operations	1: Extract Method or Extract Class
Improve error handling definition	1: Replace Error Code With Exception
	1: Convert Procedural Design to Object and Replace Conditional with Polymorphism 2: Inline Class
Improve business objects definition	3: Extract Class and Extract Subclass and Extract Superclass and Collapse Hierarchy 4: Remove Control Flag and Remove Parameter 5: Replace Type Code with Class and Replace Type Code with Subclasses
Expose shared programs as services	1: Extract Method or Extract Class
Improve names and comments	1: Rename Method or Preserve Whole Object or Introduce Parameter Object or Replace Parameter with Explicit Methods
Improve service operations cohesion	1: Inline Class and Rename Method 2: Move Method or Move Class

The proposed guidelines associate a SOA frontier improvement opportunity (Table 2, first column) with one or more OO refactorings, which are arranged in sequences of optional or mandatory refactoring combinations (Table 2, second column). Moreover, for “*Improve business objects definition*” and “*Improve service operations cohesion*”, the associated refactorings comprise more than one step. Hence, at each individual step analysts should apply the associated refactorings combinations as explained.

Regarding how to apply OO refactorings, it depends on how the WSDL documents of the SOA frontier have been built. Broadly, there are two main approaches to build WSDL documents, namely *code-first* and *contract-first* [9]. Code-first refers to automatically extract service interfaces from their underlying implementation. On the other hand, with the contract-first approach, developers should first define service interfaces using WSDL, and supplying them with implementations afterwards. Then, when the WSDL documents of a SOA frontier have been implemented under code-first, the proposed guidelines should be applied on the outermost components of the services implementation. Instead, when contract-first has been followed, the proposed OO refactorings should be applied directly on the WSDL documents.

For instance, to remove redundant operations from a code-first Web Service, developers should apply the “Extract Method” or the “Extract Class” refactorings on the underlying class that implements the service. In case of a contract-first Web Service, by extracting an operation or a port-type from the WSDL document of the service, developers apply the “Extract Method” or the “Extract Class” refactorings, but developers should also update service implementations for each modified WSDL document.

To sum up, in this section we presented an approach to automatically suggest how to improve the SOA frontier of a migrated system. This approach has been designed for reducing the costs of supporting an indirect migration attempt, while achieving a better SOA frontier than with a direct migration one. In this sense, the next section provides empirical evidence on the SOA frontier quality achieved by modernizing a legacy system using the direct, indirect, and this proposed approach to migration.

3 Evaluation

We have compared the service frontier quality achieved by direct and indirect (i.e. manual approaches), and our software-assisted migration (semi-automatic) by migrating a portion of a real-life COBOL system comprising 32 programs (261,688 lines of source code) accessing a database of around 0.8 Petabytes. Three different service frontiers were obtained: “Direct Migration”, “Indirect Migration”, and “Assisted Migration”. The comparison methodology consisted of analyzing:

- *Classical metrics*: We employed traditional software metrics, i.e. lines of code (LOC), lines of comments, and number of offered operations in the service frontiers. In this context higher values means bigger WSDL documents, which compromises clarity and thus consuming services is more difficult to application developers.
- *Data model*: Data model management is crucial in *data-centric* software systems such as the one under study. We analyzed the data-types produced by each migration approach to get an overview of data-type quality within the service frontiers. For example, we have analyzed business object definitions reuse by counting repeated data-type definitions across WSDL documents.
- *Anti-patterns*: We used the set of service interface metrics for measuring WSDL *anti-patterns* occurrences described in WSDL [8]. We used anti-patterns occurrences as an inverse quality indicator (i.e. fewer occurrences means better WSDLs).
- *Required Effort*: We used classic time and human resources indicators.

Others non-functional requirements, such as performance, reliability or scalability, have not been considered since we were interested in WSDL document quality only.

3.1 Classical Metrics Analysis

As Table 3 (top) shows, the Direct Migration data-set comprised 32 WSDL documents, i.e. one WSDL document per migrated program. Contrarily, the Indirect Migration and Assisted Migration data-sets had 7 WSDL documents and 16 WSDL documents, respectively. Having less WSDL documents means that several operations were grouped

Table 3. Classical metrics and data model analysis: Obtained results

Frontier	# of files		Total operations	Average LOC	
	WSDL	XSD		Per file	Per operation
Direct migration	32	0	39	157	129
Indirect migration	7	1	45	495	88
Assisted migration	16	1	41	235	97

Data-set	Defined data-types	Definitions per data-type	Unique data-types
		(less is better)	(more is better)
Direct Migration	182	1.29 (182/141)	141 (73%)
Indirect Migration	235	1.00 (235/235)	235 (100%)
Assisted Migration	191	1.13 (191/169)	169 (88%)

in the same WSDL document, which improves cohesion since these WSDL documents were designed to define functional related operations. Another advantage observed in the Indirect Migration and Assisted Migration data-sets over the Direct Migration data-set was the existence of an XSD file for sharing common data-types.

Secondly, the number of offered operations was 39, 45, and 41 for Direct Migration, Indirect Migration, and Assisted Migration frontiers. Although there were 32 programs to migrate, the first frontier had 39 operations because one specific program was divided into 8 operations. This program used a large registry of possible search parameters plus a control couple to select which parameters to use upon a particular search. After migration, this program was further wrapped with 6 operations with more descriptive names each calling the same COBOL program with a different control couple.

The second and third frontiers generated even more operations for the same 32 programs. This was mainly caused as *functionality disaggregation* and *routine servification* were performed during frontier generation. The former involves mapping COBOL programs that returned too many output parameters with various purposes to several purpose-specific service operations. Furthermore, the latter refers to exposing as services “utility” programs called by many other programs. Then, what used to be COBOL internal routines also become part of the SOA frontier.

Finally, although the Indirect Migration frontier had the highest LOC per file, it also had the lowest LOC per operation. The Assisted Migration frontier had a slightly higher LOC per operation than the Indirect Migration frontier. In contrast, the LOC per operation of the Direct Migration frontier was twice as much as that of the other two frontiers. Then, more code has to be read by consumers in order to understand what an operation does and how to call it, and hence WSDL documents are more cryptic.

3.2 Data Model Analysis

Table 3 (bottom) quantitatively illustrates data-type definition, reuse and composition in the three frontiers. The Direct Migration frontier contained 182 different data-types, and 73% of them were defined only once. In contrast, there were not duplicated data-types for the WSDL documents of the Indirect Migration frontier. Concretely, 104 data-types represented business objects, including 39 defined as simple XSD types (mostly

enumerations) and 65 defined as complex XSD types. Finally, 131 extra data-types were definitions introduced to be compliant with the Web Service Interoperability standards (<http://www.ws-i.org/Profiles/BasicProfile-1.1.html>), which define rules for making Web Services interoperable among different platforms. Last but not least, 191 data-types were defined in the Assisted Migration frontier, including 118 definitions in the form of business objects (34 simple XSD types + 84 complex XSD types), and 73 definitions for WS-I compliance reasons.

With respect to data-type repetitions, the Direct Migration frontier included 182 data-type definitions of which 141 were unique. This means that 23% of the definitions were not necessary and could be substituted by other semantically compatible data-type definitions. The Indirect Migration frontier, on the other hand, included 235 *unique* data-types. The Assisted Migration frontier had 191 data-types, and 169 of them were unique. Then, our tool generated WSDL documents almost as good as the ones obtained after indirectly migrating the system completely by hand. Overall, the average data-type definitions per effective data-type across WSDL documents in the frontiers were 1.29 (Direct Migration), 1 (Indirect Migration), and 1.13 (Assisted Migration).

Moreover, the analyzed WSDL frontiers contained 182, 104, and 118 different definitions of business object data-types, respectively. The Indirect Migration frontier had fewer data-type definitions associated to business objects (104) than the Direct Migration frontier (182) and the Assisted Migration frontier (118), and therefore a better level of data model reutilization and a proper utilization of the XSD complex and element constructors for WS-I related data-types. However, the Assisted Migration frontier included almost the same number of business objects than the Indirect Migration frontier, which shows the effectiveness of the data-type derivation techniques of our tool.

Figure 2 illustrates how the WSDL documents of the Indirect Migration and Assisted Migration frontiers reused the data-types. The Direct Migration frontier was left out because its WSDL documents did not share data-types among them. Unlike the Assisted Migration graph, the Indirect Migration frontier has a reuse graph without *islands*. This is because using our tool is not as good as exhaustively detecting candidate reusable data-types by hand. Nevertheless, only 2 services were not connected to the bigger graph, thus our tool adequately exploits data-type reuse.

3.3 Anti-pattern Analysis

We performed an anti-pattern analysis in the WSDL documents included in the three frontiers. We found the following anti-patterns in at least one of the WSDL documents:

- Inappropriate or lacking comments [11] (AP_1): A WSDL operation has no comments or the comments do not effectively describe its purpose.
- Ambiguous names [5] (AP_2): WSDL operation or message names do not accurately represent their intended semantics.
- Redundant port-types (AP_3): A port-type is repeated within a WSDL document, usually in the form of one port-type instance per binding type (e.g. HTTP or SOAP).
- Enclosed data model (AP_4): The data model in XSD describing input/output data-types are defined within a WSDL document instead of separate XSD files, which makes data-type reuse across several Web Services very difficult.

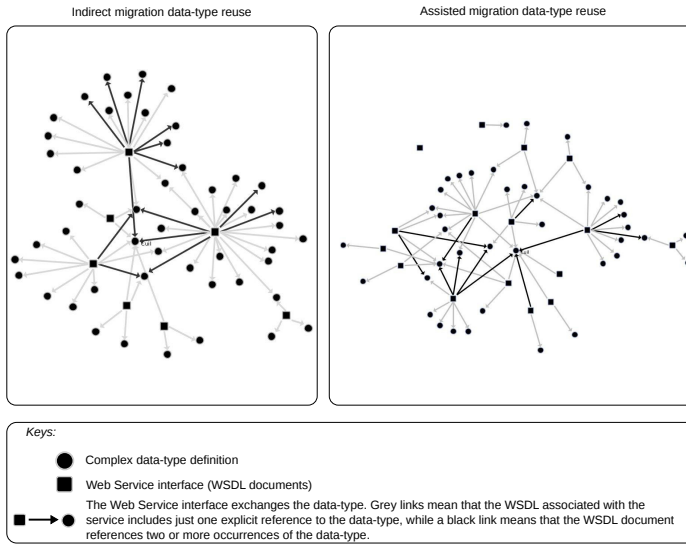


Fig. 2. Data-type reuse in the Indirect Migration and Assisted Migration frontiers

- Undercover fault information within standard messages [6] (AP_5): Error information is returned using output messages rather than built-in WSDL *fault* messages.
- Redundant data models (AP_6): A data-type is redundantly defined in a document.
- Low cohesive operations in the same port-type (AP_7): Occurs in services that place operations for checking service availability (e.g. “ping”, “isAlive”) of the service and operations related together with its main functionality into a single port-type.

Table 4 summarizes the outcomes of the analysis. When an anti-pattern affected only a portion of the WSDL documents in a frontier, we analyzed which is the difference between these WSDL documents and the rest of the WSDL documents in the same frontier. Hence, the inner cells present under which circumstances the former situation applies. Since spotting some of the anti-patterns (e.g. AP_1 and AP_2) is inherently subjective [7], we performed a peer-review methodology to prevent biases.

The Direct Migration frontier was affected by more anti-patterns than the Assisted Migration frontier, while the Indirect Migration frontier was free from anti-patterns. The first two rows describe anti-patterns that impact on services comments/names. These anti-patterns affected the Direct Migration frontier since all WSDL documents included in it were derived from code written in COBOL, which does not offer a standard way to indicate from which portions and scope of a code existing comments can be extracted and reused. Besides, COBOL names have length restrictions (e.g. up to 4 characters in some flavors). Therefore, names in the resulting WSDL documents were too short and difficult to be read. In contrast, these anti-patterns affected WSDL documents in Assisted Migration frontier only for those COBOL programs using control couples, because properly naming and commenting such couples is a complex task [12].

Table 4. Anti-patterns analysis: Obtained results

Anti-pattern/Frontier	Direct Migration	Indirect Migration	Assisted Migration
AP_1	Always	Never	When the original COBOL programs use control couples
AP_2	Always	Never	When the original COBOL programs use control couples
AP_3	When supporting several protocols	Never	Never
AP_4	Always	Never	Never
AP_5	Always	Never	Never
AP_6	When two operations use the same data-type	Never	Never
AP_7	Never	Never	When several related programs link to non-related operations

The third row analyzes the anti-pattern that ties abstract service interfaces (WSDL port-types) to concrete implementations (WSDL bindings), and as such hinders black-box reuse [8]. We observed that this anti-pattern was caused by the WSDL generation tools supporting the migration process that resulted in the Direct Migration frontier. Unless properly configured, these tools by default produce redundant port-types when deriving WSDLs from COBOL programs. Likewise, the fourth row describes an anti-pattern that is generated by this tool as well as many similar tools, which involves forcing data models to be included within the generated WSDL documents, making cross-WSDL data-type reuse difficult. Alternatively, neither the Indirect Migration nor the Assisted Migration frontiers were affected by these two anti-patterns.

The anti-pattern described in the fifth row of the table deals with errors being transferred as part of output messages, which for the Direct Migration frontier resulted from the original COBOL programs that used the same data output record for returning both output and error information. In contrast, the WSDL documents of the Indirect Migration frontier and the Assisted Migration frontier had a proper designed error handling mechanism based on standard WSDL *fault* messages.

The anti-pattern described in the sixth row is related to badly designed data models. Redundant data models usually arise from limitations or bad use of WSDL generation tools. Therefore, this anti-pattern only affected the Direct Migration frontier. Although there was not intra WSDL data-type repetition, the Assisted Migration frontier suffered from inter WSDL repeated data-types. For example, the `error` data-type—which consists of a fault code, a string (brief description), a source, and a description—was repeated in all the Assisted Migration WSDL documents because the data-type was derived several times from the different various programs. This problem did not affect the Indirect Migration frontier since its WSDL documents were manually derived by designers after having a big picture of the legacy system.

The last anti-pattern stands for having no semantically related operations within a WSDL port-type. This anti-pattern neither affected the Direct Migration nor the Indirect Migration frontier because in the former case each WSDL document included only one operation, whereas in the latter case WSDL documents were specifically designed to group related operations. However, our approach is based on an automatic heuristic

that selects which operations should go to a port-type. In our case study, we found that when several related operations used the same unrelated routines, such as text-formatting routines, our assisted approach to migration suggested that these routines were also candidate operations for that service. This resulted in services that had port-types with several related operations but also some unrelated operations.

3.4 Required Effort Analysis

In terms of manpower, it took 1 day to train a developer on the method and tools used for building the Direct Migration frontier. Then, a trained developer migrated one transaction per hour. Thus, it only took 5 days for a trained developer to build the 32 WSDL documents. Instead, building the Indirect Migration frontier demanded one year plus one month, and 8 software analysts, and 3 specialists for migrating the same 32 transactions, from which 5 months were exclusively dedicated to build its WSDL documents.

We also have empirically assessed the time needed to execute our semi-automatic heuristics. The experiments have been run on a 2.8 GHz QuadCore Intel Core i7 720QM machine, 6 Gb RAM, running Windows 7 on a 64 bits architecture. To mitigate noise introduced by underlying software layers and hardware elements, each heuristic has been executed 20 times and the demanded time was measured per execution. Briefly, the average execution time of an heuristic was 9585.78 ms, being 55815.15 ms the biggest achieved response time, i.e. the “Detect semantically similar services and operations” was the most expensive heuristic in terms of response time.

Furthermore, we have assessed the time demanded for manually applying the OO refactorings proposed by the approach on the Direct Migration frontier. To do this, one software analyst with solid knowledge on the system under study was supplied with the list of OO refactorings produced by the approach. It took two full days to apply the proposed OO refactorings. In this sense, the approach suggested to “Expose 6 shared programs as services”, “Remove 7 redundant operations”, “Improve the cohesion of 14 services”, and to “Improve error handling definition”, “Improve names and comments”, and “Improve business objects definition” from all the migrated operations. It is worth noting that OO refactorings have been applied at the interface level, i.e. underlying implementations have not been accommodated to interface changes. The reason to do this was we only want to generate a new SOA frontier and then compare it with the ones generated by the previous two migration attempts. Therefore, modifying interfaces implementation, which would require a huge development and testing effort, would not contribute to assessing service interfaces quality.

4 Related Work

Migration of mainframe legacy systems to newer platforms has been receiving lots of attention as organizations have to shift to distributed and Web-enabled software solutions. Different approaches have been explored, ranging from wrapping existing systems with Web-enabled software layers, to 1-to-1 automatic conversion approaches for converting programs in COBOL to 4GL. Therefore, current literature presents many related experience reports. However, migrating legacy systems to SOA, while achieving high-quality service interfaces instead of just “webizing” the systems, is an incipient research topic.

Harry Sneed has been simultaneously researching on automatically converting COBOL programs to Web Services [13] and measuring service interfaces quality [14]. In [13] the author presents a tool for identifying COBOL programs that may be *servified*. The tool bases on gathering code metrics from the source to determine program complexity, and then suggests whether programs should be wrapped or re-engineered. As such, programs complexity drives the selection of the migration strategy. As reported in [13], Sneed plans to inspect resulting service interfaces using a metric suite of his own, which comprises 25 quantity, 4 size, 5 complexity and 5 quality metrics.

In [15] the authors present a framework and guidelines for migrating a legacy system to SOA, which aims at defining a SOA frontier having only the “optimal” services and with an appropriate level of granularity. The framework consists of three stages. The first stage is for modeling the legacy system main components and their interactions using UML. At the second stage, service operations and services processes are identified. The third stage is for aggregating identified service elements, according to a predefined taxonomy of service types (e.g. CRUD Services, Infrastructure services, Utility services, and Business services). During the second and third stages, software analysts are assisted via clustering techniques, which automatically group together similar service operations and services of the same type.

5 Conclusions and Future Work

Organizations are often faced with the problem of legacy systems migration. The target paradigm for migration commonly used is SOA since it provides interoperability and reusability. However, migration to SOA is in general a daunting task.

We proposed a semi-automatic tool to help development teams in migrating COBOL legacy systems to SOA. Our tool comprises 10 heuristics that detect bad design and implementation practices in legacy systems, which in turn are related to some early code refactorings so that services in the final SOA frontier are as clear, legible and discoverable as possible. Through a real-world case study, we showed that our approach dramatically reduced the migration costs required by indirect migration achieving at the same time a close service quality. In addition, our approach produced a SOA frontier much better in terms of service quality than that of “fast and cheap” approach to migration (i.e. direct migration). The common ground for comparison and hence assessing costs and service quality was some classical software engineering metrics, data-type related metrics, and a catalog of WSDL anti-patterns [8] that hinder service reusability.

At present, we are refining the heuristics of our tool to improve their accuracy. Second, we are experimenting with an RM-COBOL system comprising 319 programs and 201,828 lines of code. In this line, we will investigate whether is possible to adapt all the evidences heuristics to be used with other COBOL platforms. Lastly, even when direct migration has a negative incidence in service quality and WSDL anti-patterns, there is recent evidence showing that many anti-patterns are actually introduced by the WSDL generation tools used during migration [16]. Our goal is to determine to what extent anti-patterns are explained by the approach to migration itself, and how much of them depend on the WSDL tools used.

References

1. Rodriguez, J.M., Crasso, M., Mateos, C., Zunino, A., Campo, M.: Bottom-up and top-down COBOL system migration to Web Services: An experience report. In: IEEE Internet Computing (2011) (to appear)
2. Bichler, M., Lin, K.-J.: Service-Oriented Computing. *Computer* 39(3), 99–101 (2006)
3. Erickson, J., Siau, K.: Web Service, Service-Oriented Computing, and Service-Oriented Architecture: Separating hype from reality. *Journal of Database Management* 19(3), 42–54 (2008)
4. Li, S.-H., Huang, S.-M., Yen, D.C., Chang, C.-C.: Migrating legacy information systems to Web Services architecture. *Journal of Database Management* 18(4), 1–25 (2007)
5. Brian Blake, M., Nowlan, M.F.: Taming Web Services from the wild. *IEEE Internet Computing* 12(5), 62–69 (2008)
6. Beaton, J., Jeong, S.Y., Xie, Y., Jack, J., Myers, B.A.: Usability challenges for enterprise service-oriented architecture APIs. In: IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 193–196 (September 2008)
7. Rodriguez, J.M., Crasso, M., Zunino, A., Campo, M.: Automatically Detecting Opportunities for Web Service Descriptions Improvement. In: Cellary, W., Estevez, E. (eds.) *Software Services for e-World. IFIP AICT*, vol. 341, pp. 139–150. Springer, Heidelberg (2010)
8. Rodriguez, J.M., Crasso, M., Zunino, A., Campo, M.: Improving Web Service descriptions for effective service discovery. *Science of Computer Programming* 75(11), 1001–1021 (2010)
9. Mateos, C., Crasso, M., Zunino, A., Campo, M.: Separation of concerns in service-oriented applications based on pervasive design patterns. In: *Web Technology Track (WT) - 25th ACM Symposium on Applied Computing (SAC 2010)*, pp. 2509–2513. ACM Press (2010)
10. Fowler, M.: *Refactorings in Alphabetical Order* (1999)
11. Fan, J., Kambhampati, S.: A snapshot of public Web Services. *SIGMOD Rec.* 34(1), 24–32 (2005)
12. Yourdon, E., Constantine, L.L.: *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Prentice-Hall, Inc., Upper Saddle River (1979)
13. Sneed, H.: A pilot project for migrating COBOL code to Web Services. *International Journal on Software Tools for Technology Transfer* 11, 441–451 (2009) 10.1007/s10009-009-0128-z
14. Sneed, H.: Measuring Web Service interfaces. In: *12th IEEE International Symposium on Web Systems Evolution*, pp. 111–115 (September 2010)
15. Alahmari, S., Zaluska, E., De Roure, D.: A service identification framework for legacy system migration into SOA. In: *Proceedings of the IEEE International Conference on Services Computing*, pp. 614–617. IEEE Computer Society (2010)
16. Mateos, C., Crasso, M., Zunino, A., Coscia, J.L.O.: Detecting WSDL bad practices in code-first Web Services. *International Journal of Web and Grid Services* 7(4), 357–387 (2011)

1+1 Protection of Overlay Distributed Computing Systems: Modeling and Optimization

Krzysztof Walkowiak¹ and Jacek Rak²

¹ Wroclaw University of Technology, Wybrzeze Wyspianskiego 27,
PL-50-370 Wroclaw, PL

² Gdansk University of Technology, G. Narutowicza 11/12, PL-80-233 Gdansk, PL
krzysztof.walkowiak@pwr.wroc.pl, jrak@pg.gda.pl

Abstract. The development of the Internet and growing amount of data produced in various systems have triggered the need to construct distributed computing systems required to process the data. Since in some cases, results of computations are of great importance, (e.g., analysis of medical data, weather forecast, etc.), survivability of computing systems, i.e., capability to provide continuous service after failures of network elements, becomes a significant issue. Most of previous works in the field of survivable computing systems consider a case when a special dedicated optical network is used to connect computing sites. The main novelty of this work is that we focus on overlay-based distributed computing systems, i.e., in which the computing system works as an overlay on top of an underlying network, e.g., Internet. In particular, we present a novel protection scheme for such systems. The main idea of the proposed protection approach is based on 1+1 protection method developed in the context of connection-oriented networks. A new ILP model for joint optimization of task allocation and link capacity assignment in survivable overlay distributed computing systems is introduced. The objective is to minimize the operational (OPEX) cost of the system including processing costs and network capacity costs. Moreover, two heuristic algorithms are proposed and evaluated. The results show that provisioning protection to all tasks increases the OPEX cost by 110% and 106% for 30-node and 200-node systems, respectively, compared to the case when tasks are not protected.

Keywords: distributed computing systems, protection, survivability, optimization.

1 Introduction

Distributed computing systems are developed to process tasks requiring huge processing power, which is not obtainable on a single machine. There are two major categories of such systems: Grids and P2P (Peer-to-Peer) computing systems, also called public resource computing systems. Grids are issued by organizations and institutions, and contain a small number (usually up to hundred) of machines connected by means of network links of high efficiency. P2P computing systems

consist of a number of small machines (e.g., PC or Macintosh computers, gaming consoles) utilizing access links like Fast Ethernet, DSL, WiFi, HSPA, etc. Using a simple software, each machine can be registered to a selected computing project and offer spare computing resources. This approach is much simpler than Grid structures, since the only need is to provide a suitable software and to manage tasks and results (in case of Grid systems, also physical machines must be maintained) [1]-[4].

Efficiency of distributed computing systems may be remarkably decreased by failures of network elements (i.e., nodes, or links) most frequently caused by human errors (e.g., cable cuts), or forces of nature (e.g., earthquakes, hurricanes). Therefore, network survivability, i.e., ability to provide the continuous service after a failure [5], becomes a crucial issue. Most commonly, it is assured by means of additional (backup) resources (e.g., transmission links/paths, computing units) used after the failure affecting the components of the main communication path (called working path) as the main network resources of task processing.

It is worth noting that in order to provide protection against failures of nodes/links, backup paths by definition should not have any common transit node/link with their working paths, accordingly. In general, survivability approaches can be classified as either proactive (e.g. protection scheme based on reserving the backup resources in advance) or reactive methods (using dynamic restoration to find backup paths only after a failure) [6]. Finally, based on the scope of a backup path protection, they can be divided into path, segment or link protection/restoration schemes [7]–[9], as shown in Fig. 1. In the latter two schemes (Figs. 1b and 1c), there is more than one backup path protecting a given working path.

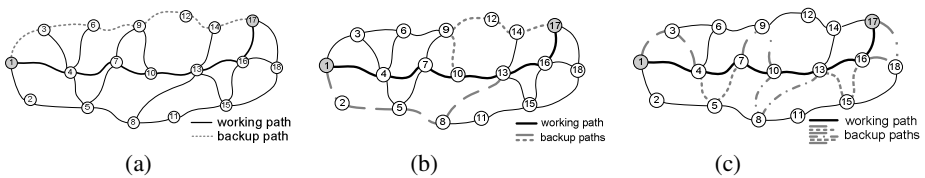


Fig. 1. Examples of a path- (a), segment- (b), and link-protection/restoration scheme

In this paper, we focus on modeling and optimization of survivable distributed computing systems. In particular, we use the proactive approach based on path protection scheme that additionally conforms to the 1+1 method applied in connection-oriented computer networks. The main idea of this method is to transmit information via two disjoint paths at the same time (“+” is used to denote parallel transmission). Therefore, in normal operating state (i.e., with no failures) information is concurrently sent/received to/from two computing nodes. After a failure, one of the considered disjoint paths remains operational, implying that transmission of information is provided with no violation at all.

The investigated computing system is to be protected against a failure that leads to a situation when the results produced by one of the nodes are not transported to the destination node(s). This can be caused by a wide range of breakdowns including both networking issues (e.g., access link failure, backbone network link failure, backbone

node failure, etc.) and processing issues (node hardware failure, power outage, etc.). To provide protection against such outages, tasks are allocated to primary and backup nodes. The objective is to allocate tasks to computing nodes and assign access link capacity to each computing node in order to minimize the operational (OPEX) cost and satisfy several constraints including link and node capacity constraints, as well as additional protection requirements.

The key novelty of this work is that we consider a survivable distributed system working as an overlay network, i.e., a network that is built on top of an existing underlying network providing basic networking functionalities including routing and forwarding. This is in contrast to previous papers covering issues of survivable computing systems that have been focused on systems using a dedicated optical network deployed to connect computing sites to construct the Grid systems, e.g., [10]-[13]. In this overlay structure, underlying network can be based on either wired or even wireless communications (the latter one realized e.g., according to 802.11s standard of wireless mesh networks with highly directional antennas providing mutually non-interfering transmission links).

The main contributions of the paper are as follows: (i) a new protection approach to provide survivability of overlay distributed computing systems, (ii) introduction of an ILP model related to survivable overlay distributed computing systems, (iii) presentation of two heuristic algorithms proposed to solve the formulated problem, (iv) results of numerical experiments showing the performance of heuristic algorithms compared against the optimal results given by CPLEX 11.0, and the main characteristics of survivable distributed computing systems. To the best of our knowledge, no similar method exists in the literature.

The rest of the paper is organized in the following way. Information on related works is provided in Section II. Section III describes a novel protection concept for distributed computing systems. In Section IV, a new ILP model is presented. Section V includes description of heuristic algorithms. Simulation assumptions and results are discussed in Section VI.

2 Related Work

Survivability aspects have been extensively studied in the literature in the context of optical WDM networks, i.e., to protect the flows against failures of nodes/links [5], [14]-[15]. In particular, majority of these papers refer to unicast (i.e., one-to-one transmission, where working and backup paths of a given demand have the same source/destination nodes). Among them, we may distinguish papers on differentiated scopes of protection (e.g., [16]-[18]), or differentiated protection with respect to various classes of service [19]-[20].

The authors of [21] introduced a model of an overlay distributed computing system, which may be used for multiple classifier systems. An ILP model is formulated. The objective is to optimize allocation of tasks to computing nodes to minimize the OPEX cost related to processing and transfer of data. An efficient heuristic algorithm based on the GRASP approach is developed and examined. It

should be noted that the model considered in this paper uses similar assumptions to [21]. However, contrary to [21], additional survivability requirements are included as well as dimensioning of link capacity is considered here.

Survivability of distributed computing systems (in particular including protection of computing units), is a relatively new topic, and only a few proposals exist in the literature. However, most of them assume that the computing system uses a dedicated optical network, e.g., [10]-[13]. In particular, the authors of [10] extended the problem of providing survivability in optical networks (which is a well-researched topic) by incorporating Grid computing assumptions. They introduced an approach to provide resiliency against failures of network elements, as well as computing units. The paper [11] addresses the issue of optimization of optical Grids considering combined dimensioning and workload scheduling problem with additional survivability requirements. The Divisible Load Theory is applied there to solve the scalability problem and compare non-resilient lambda Grid dimensioning with the dimensions needed to survive single-resource failures. The authors of [12] proposed to extend the existing Shared Path protection scheme by incorporating the anycast paradigm characteristic for Grids. Consequently, the backup path from the source can be terminated in a node other than the origin server node, i.e., it is allowed to relocate the job to another server location. In [13], resilient optical Grids are considered. The authors examined how to maximize the grade of services for given transport capacities, while maximizing the protection level, i.e., against single link vs. single link and node (including server node) failures. A large scale optimization model is solved with help of Column Generation (CG) technique.

To the best of our knowledge, there are currently no papers considering the survivability of distributed computing systems working in overlay mode.

3 Survivability of Distributed Computing Systems

3.1 Distributed Computing System Architecture

In this section, we introduce the architecture of a distributed computing system addressed in this paper. Our assumptions follow from previous works on this topic (e.g., [2]-[3],[22]), and real systems like Seti@Home using BOINC framework [23].

Distributed computing systems can be built using either a network model, or an overlay model. In the former case, a dedicated network connecting computing nodes is created, i.e., network links between these nodes are established and dimensioned. The latter case assumes that computing nodes form a virtualized network over the underlying physical network (mostly the Internet), which is used to provide network connectivity between computing nodes with required Quality of Service guarantees. Overlay routing allows for a flexible network operation, e.g., any failure encountered at the overlay system layer can be fixed by the overlay network itself (without the need to cooperate with the underlying physical layer).

In this paper, we focus on the overlay case. However, presented model can be easily augmented to consider additional constraints of the network model. In our

architecture, it is assumed that computing nodes (peers) constituting the system are connected through the Internet (an overlay network). Moreover, a full mesh structure is assumed, i.e., each peer can be connected to any other peer. Thus, there is no need to provide additional routing. Physically, the direct connectivity between nodes is provided by the underlying Internet layer. In this paper, we do not consider emulation of an additional substrate (e.g., Distributed Hash Table - DHT in P2P networks). The motivation behind this decision is that the predicted number of participants of computing systems (e.g., several hundred) is not large.

The considered distributed computing system is designed to process various computational tasks. The workflow of the computing system is as follows. The input data of the task is transferred from the source node to one or more computing nodes that process the data. Next, the output data (results of computations) is sent from the computing node to one or more destination nodes. We assume that computational tasks are long-lived, i.e., they are established for a relatively long time (days, or weeks). Examples of such systems include: scientific experimental systems (e.g., Large Hadrons Collider), classification systems (e.g., anti-spam filters), data mining (e.g., sales forecasts), weather forecast systems. Therefore, we do not consider here – as in many other works (e.g., in [2]) the time dependency of each task (starting time, completion time, etc.). Consequently, the input and output data associated with the task is continuously generated and transmitted. Therefore, computational and network resources can be allocated in the system according to the offline optimization process results.

Each computing node is connected to the overlay by an access link with a specified capacity that is used only to transmit the data related to task processing. More precisely, each node can download two kinds of data: task input data (if the particular node is selected for processing of the considered task), and task output data (if the particular node is the destination node of the considered task). Similarly, each node can upload two kinds of data: task input data (if the particular node is the source node of the considered task), and output data (if the particular node is selected for processing of the considered task).

Each node is equipped with various kinds of devices that can be applied for computing (e.g., computers, clusters, game consoles, etc.). Each node has a limited processing power defined by the number of uniform computational tasks that the node can calculate in one second. Summarizing, the considered model of a computing system assumes that each node has two kinds of resources: network (access link to the overlay), and processing power. In this paper, we focus on static optimization of the system – the objective is to allocate tasks to computing nodes in order to minimize the OPEX cost of the computing system including expenses related to processing (task computation), and network (access link).

A simple example to illustrate the distributed computing system architecture is presented in Fig. 2. The considered system includes five computing nodes denoted as A, B, C, D, and E. Nodes are connected to the overlay network. There are three tasks to be processed. Node A is the source node of all tasks. Therefore, it stores input data related to the tasks (green rectangles labeled i_1 , i_2 and i_3). Orange rectangles labeled p_1 , p_2 , and p_3 denote the places where a particular task is processed,

i.e., node B processed task 1, node A processed task 2, and node E processed task 3. Red rectangles labeled $\circ 1$, $\circ 2$, and $\circ 3$ denote results of computations related to tasks 1, 2 and 3, respectively. Nodes B and D are destinations of all tasks. Moreover, in this figure we show network flows generated to deliver the input and output data. Solid line denotes the flow of input data. Green circle with a number inside shows the indices of tasks the input data is related to. For instance, since node B is selected to compute task 2, the input data of this task is sent from node A to node B. Dotted line shows the flow of output data. Again, the numbers in red circles indicate task indices the data belongs to. For instance, since node E processes task 3, this node uploads the output data to nodes B and D (destination nodes). Notice that the input data related to task 2 is not sent from node A, as node A itself processes the task. Analogously, the output data of task 1 calculated in node B is uploaded only to node D, as the second destination node is B.

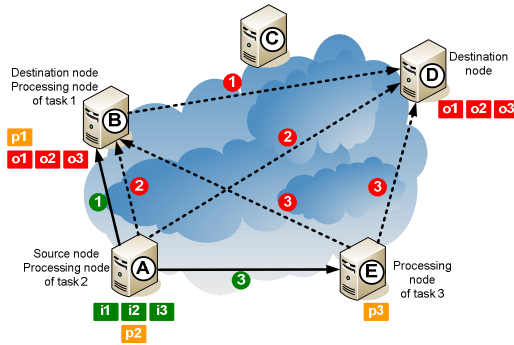


Fig. 2. Example of a distributed computing system

3.2 Protection Approach

The growing need to process a large amount of data explains the fact that various distributed computing systems including P2P systems have been gaining much attention in recent years. Moreover, many computational tasks processed in distributed systems are of great importance and need execution guarantees, e.g., medical applications, business analysis, weather forecasts, etc. However, considered computing systems operate on an overlay network. Therefore – similar to communication networks – they are subject to various unintentional failures caused by natural disasters (hurricanes, earthquakes, floods, etc.), overload, software bugs, human errors, and intentional failures caused by maintenance action or sabotage [15]. Such failures influence network infrastructure connecting computing nodes (e.g., access link failure, underlying physical network link failure, etc.). Additionally, elements of distributed computing systems devoted to data processing are also subject to various breakdowns (e.g., hardware failure, power outage, software bug, etc.). Therefore, in distributed computing systems, to provide guarantees on computational tasks completion, execution and delivery of all required results need to be enhanced with some survivability mechanisms.

As a failure scenario, we consider a failure that leads to a situation when the results to be obtained at one of the computing nodes are not delivered to the requesting destination nodes (e.g., access link failure, backbone network link failure, backbone node failure, etc., and processing issues including node hardware failure, power outage, etc.). To protect the distributed computing system against this kind of a failure, we propose to apply a similar approach as in connection-oriented networks, i.e., 1+1 protection developed in the context of Automatic Protection Switching (APS) networks [5]. In our approach, there is one primary (working) system and a completely reserved backup system. In a fully operational state (with no failures), backup system transmits the copied signal. The main advantage of the 1+1 approach is a very fast reaction after the failure. However, the disadvantage is a relatively large cost following from the system redundancy. Our idea is to assign to each computational task two computing nodes: primary and backup. Both nodes simultaneously process the same input data and next send results to all destination nodes. Compared to the architecture presented in the previous section, the general structure of the system is not changed. However, the need to process an additional copy of the task results in extra network traffic and requires more computational resources.

In Fig. 3, we show a survivable computing system using the same example as in Fig. 2. To address the survivability requirements, each of three tasks is processed at two separate nodes – primary tasks are marked with rectangles p1, p2, and p3, while backup tasks are labelled b1, b2 and b3, respectively. For instance, task 1 is processed at nodes B and C. As mentioned above, duplication of tasks generates much more network traffic compared to the example from Fig. 2. For instance, node A uploads two copies of task 1 and 3 input data, and one copy of task 2 input data.

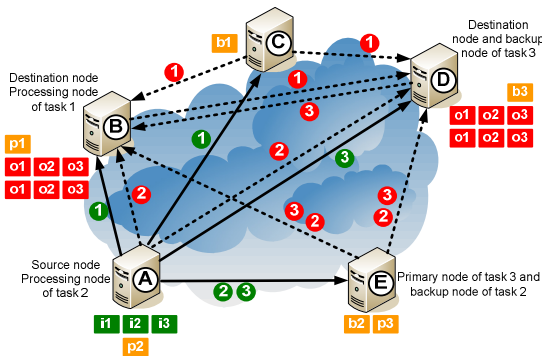


Fig. 3. Example of a survivable distributed computing system

4 Integer Programming Model

In this section, we present our Integer Programming model of the survivable distributed computing system. The motivation is to use the introduced model to apply

an offline optimization algorithm (i.e., branch-and-cut) to find the optimal results that can be used as the lower bounds for other solution approaches (e.g., heuristics). Additionally, in many cases, distributed computing systems are static (e.g., in scientific computational projects), and results of offline optimization can be applied there to improve the system performance.

The considered computing system – as described in the previous section – is constructed as a set of computing nodes (individual computers or clusters) – denoted as $v = 1, 2, \dots, V$. An overlay network is used to connect all computing nodes. As stated in [24]-[25], the physical network underlying the overlay is assumed to be overprovisioned. Therefore, we do not consider in this model the capacity constraints of the underlying physical network – the only network bottlenecks of the overlay system are access links. Each node v is assigned a maximum processing rate p_v (i.e., each node is already equipped with some computers, and p_v denotes the number of uniform computational tasks that node v can calculate in one second). For each node v , ψ_v denotes the OPEX cost related to processing of one uniform task at node v . The cost ψ_v is given in euro/month and includes all expenses necessary to process the uniform computational tasks, e.g., energy, maintenance, administration. The processing cost ψ_v can be different for various nodes, since differentiated costs related e.g., to energy, or maintenance, etc., apply to particular node locations.

Another important parameter of the node is the capacity of the access link. Integer variable z_v denotes the number of capacity modules allocated to the access link of node v . We assume that each node v is assigned to a particular Internet Service Provider (ISP) that offers high speed access link with basic capacity modules m_v given in Mbps (e.g., Fast Ethernet). Thus, capacity of node v access link is $m_v z_v$. Moreover, ξ_v given in euro/month denotes the whole OPEX cost related to one capacity module allocated for node v . Constant ξ_v includes the leasing cost of the capacity module paid to the ISP and all other OPEX costs like energy, maintenance, administration, etc. Since computing nodes are spread geographically in many countries with various ISPs, energy and maintenance costs, as well as module size and cost can be different for each node. Note that – if necessary – a number (more than 1) of capacity modules can be assigned to node v . Decision variable z_v is thus integer.

Network computing system is to process the tasks with indices $r = 1, 2, \dots, R$. Each task is defined by the source node producing the input data, and one or more destination nodes receiving the output data including results of computations. Constant s_{rv} is 1, if node v is the source node of task r ; 0 otherwise. Similarly, t_{rv} is 1, if node v is the destination node of task r ; 0 otherwise. Input and output data transmission rates of task r are given by constants a_r and b_r , respectively.

We consider a survivable distributed computing system, i.e., some tasks are of great importance and require special protection against potential failures. Constant α_r is used to indicate protection required by task r . Each task r that requires protection (i.e., $\alpha_r = 1$) is allocated for processing at two different nodes: a primary and a backup one. Binary variable x_{rv} is equal to 1, if task r is allocated to the primary computing node v ; 0 otherwise. Similarly, y_{rv} is 1, if task r is allocated to the backup computing node v ; 0 otherwise. Note that if a particular task r does not require protection (i.e., $\alpha_r = 0$), then the backup node is not selected.

The objective of the optimization model is to minimize the operating cost of the computing system including expenses related to network (access links) and processing. We optimize selection of access links (variable z_v), and allocation of tasks to primary and backup nodes (modeled by variables x_{rv} and y_{rv} , accordingly). It should be underlined that the OPEX cost is perceived from the perspective of the computing system, not from the perspective of the network operator. Therefore, the network cost is limited only to access links and does not include costs related to the backbone network underlying the overlay computing system.

indices

$v = 1, 2, \dots, V$ computing nodes (peers)
 $r = 1, 2, \dots, R$ computing tasks

constants

p_v maximum processing rate of node v (number of uniform tasks that node v can calculate in one second)
 a_r transmission rate of task r input data (Mbps)
 b_r transmission rate of task r output data (Mbps)
 s_{rv} equals 1, if v is the source node of task r ; 0 otherwise
 t_{rv} equals 1, if v is the destination node of task r ; 0 otherwise
 t_r number of destination nodes for task r , i.e., $t_r = \sum_v t_{rv}$
 α_r equals 1, if task r requires protection; 0 otherwise
 ψ_v OPEX cost related to processing of one uniform task in node v (euro/month)
 ξ_v OPEX cost related to one capacity module of node v (euro/month)
 m_v size of the capacity module for node v (Mbps)

variables

x_{rv} equals 1, if task r is allocated to the primary computing node v ; 0 otherwise (binary)
 y_{rv} equals 1, if task r is allocated to the backup computing node v ; 0 otherwise (binary)
 z_v capacity of node v access link expressed in terms of the number of capacity modules (non-negative integer)

objective

$$\text{minimize } C = \sum_v z_v \xi_v + \sum_r \sum_v (x_{rv} \psi_v + y_{rv} \psi_v) \tag{1}$$

constraints

$$\sum_r (x_{rv} + y_{rv}) \leq p_v \quad v = 1, 2, \dots, V \tag{2}$$

$$\sum_r (1 - s_{rv}) a_r (x_{rv} + y_{rv}) + \sum_r t_r b_r (1 - x_{rv} + \alpha_r - y_{rv}) \leq z_v m_v \quad v = 1, 2, \dots, V \tag{3}$$

$$\sum_r s_{rv} a_r (1 - x_{rv} + \alpha_r - y_{rv}) + \sum_r (t_r - t_{rv}) b_r (x_{rv} + y_{rv}) \leq z_v m_v \quad v = 1, 2, \dots, V \quad (4)$$

$$\sum_v x_{rv} = 1 \quad r = 1, 2, \dots, R \quad (5)$$

$$\sum_v y_{rv} = \alpha_r \quad r = 1, 2, \dots, R \quad (6)$$

$$(x_{rv} + y_{rv}) \leq 1 \quad r = 1, 2, \dots, R \quad v = 1, 2, \dots, V \quad (7)$$

Value of the objective function (1) reflects the OPEX cost including two elements: network cost ($\sum_v z_v \xi_v$), and processing cost ($\sum_r \sum_v (x_{rv} \psi_v + y_{rv} \psi_v)$). Notice that the processing cost comprises costs related to both primary and backup nodes. Constraint (2) follows from the fact that each computing node v has a limited processing power p_v . Therefore, each node cannot be assigned with more tasks to calculate than it can process. Since computations on primary and backup nodes are made simultaneously, the left-hand side of (2) includes variables x_{rv} and y_{rv} .

Formula (3) defines the download capacity constraint of a node access link. The left-hand side of (3) denotes flow incoming to node v and includes two elements. The former one $\sum_r (1 - s_{rv}) a_r (x_{rv} + y_{rv})$ denotes the flow related to transmission of input data for computations, i.e., node v that is selected as a primary node of task r ($x_{rv} = 1$), or a backup node of task r ($y_{rv} = 1$), must download the input data with transmission rate a_r . Only if the considered node v is the source node of task r ($s_{rv} = 1$), there is no need to transmit the input data. The latter element $\sum_r t_{rv} b_r (1 - x_{rv} + \alpha_r - y_{rv})$ refers to the output data transmission – each destination node v of task r ($t_{rv} = 1$) must download the output data of this task from both primary and backup computing node. However, if the considered node v is selected as a primary node ($x_{rv} = 1$), there is no need to download data to the primary node. Similarly, if the considered task r is protected ($\alpha_r = 1$) and the considered node v is selected as a backup node ($y_{rv} = 1$), there is no need to download data to the backup node. The right-hand side of (3) denotes the access link capacity.

Formula (4) denotes the upload capacity constraints. Analogous to (3), the left-hand side of (4) defines the flow leaving node v and includes two terms. The first one: $\sum_r s_{rv} a_r (1 - x_{rv} + \alpha_r - y_{rv})$ denotes the input data flow sent from the task r source node v ($s_{rv} = 1$) to primary and backup computing nodes. However, if the considered node v is selected as either the primary computing node ($x_{rv} = 1$), or the backup computing node ($y_{rv} = 1$) of task r , the corresponding flow is set to 0. The second term $\sum_r (t_r - t_{rv}) b_r (x_{rv} + y_{rv})$ is related to delivery of output data to all destination nodes of task r , i.e., each node v that is selected as either the primary node of task r ($x_{rv} = 1$), or the backup node of task r ($y_{rv} = 1$), must upload the output data to destination nodes of task r (denoted by t_r). If the considered node v is one of destination nodes of task r ($t_{rv} = 1$), then the flow is decreased adequately. The right-hand side of (4) denotes again the access link capacity.

Condition (5) assures that for each task, exactly one node is selected as the primary node. Similarly, (6) guarantees that if task r is to be protected ($\alpha_r = 1$), then one backup node has to be selected. Constraint (7) assures that primary and backup nodes are disjoint.

Note that in [21], an optimization problem similar to (1)–(7) is considered. The main difference is that the model in [21] did not include link capacity assignment and survivability constraints. Compared to [10]–[13], here we use an overlay network model, while in [10]–[13] the optical mesh network topology is assumed.

5 Heuristic Algorithms

Due to the complexity of the optimization problem given by formulas (1)–(7), exact methods like branch-and-cut can be used to solve only relatively small instances. Therefore, here we propose two heuristic algorithms. The first one – referred to as AlgRand – is a random method based on common characteristics of real overlay systems, which mostly run in a random manner, e.g., BitTorrent [22]. The idea is to allocate a randomly selected task r to a randomly selected node v . However, only nodes with free processing resources are taken into account. When all tasks are allocated, capacity of access links is determined in order to enable distribution of input and output data. Execution of the algorithm is repeated here for 20 times, and the final result is the minimum value of the obtained cost.

The second heuristic approach called AlgGreedy is a greedy algorithm that analyzes all tasks in a single run. Again, only feasible nodes (with free processing resources) are considered as candidate nodes. In each iteration, the cheapest allocation of task r to node v is selected according to metric c_{rv} , being the weighted sum of two elements. First, we calculate an average cost of task r to node v allocation taking into account processing cost of node v and the cost related to network capacity. More precisely, allocation of task r to node v generates flow of input data from the source node of task r to node v , and next flow of the output data from node v to all destination nodes of task r . For all these flows, network capacity must be allocated. Therefore, we estimate the capacity cost.

For instance, the cost related to the download capacity of the computing node v and task r is given by formula $\xi_r a_r / m_v$. All other costs are calculated in analogous way. The second part of metric c_{rv} is calculated in a slightly different way. The idea is to check if allocation of task r to node v triggers the need to add a new capacity module(s) to some nodes (to provide enough capacity to send input/output data). It means that in some cases, allocation of task r to node v does not require augmenting of capacity, since the already allocated resources are sufficient to send all the required data.

6 Results

In this section, we report the results of computational experiments. The goal of experiments was twofold. First, we wanted to evaluate the performance of heuristic

algorithms compared against the optimal results given by CPLEX 11.0 [26]. Second, we examined how additional survivability requirements influence the OPEX cost of the distributed computing system. Simulations were run on six distributed computing systems generated at random.

In particular, number of computing nodes was 30, capacity module was 100 Mbps, other parameters of computing nodes were uniformly distributed in the following ranges: cost of capacity module in range 120-400; processing cost in range 50-150; processing limit in range 10-40. Six sets of computing tasks were created at random with the following settings: number of tasks in range 300-700; number of destination nodes of each task in range 1-6; input and output data transmission rate in range 5-15 Mbps. Moreover, 11 configurations related to protection requirements were created with the following values of the protected tasks percentage (PTP): 0%, 10%, 20%, ..., 100%. Thus, the overall number of individual cases tests was 396 (i.e., $6 \times 6 \times 11$).

We used CPLEX solver to obtain the reference results. However, when using the default setting of the optimality gap (i.e., 0.0001), CPLEX was not able to stop calculations within one hour for one test. Therefore, we decided to set optimality gap to 0.01 – then CPLEX average execution time was about 29 seconds, while the obtained average optimality gap was 0.0087. The average running time of heuristics was 0.4 and 0.007 seconds for AlgGreedy and AlgRand, respectively. In Table I, we report the performance characteristics of heuristics compared against the optimal results returned by CPLEX.

For each algorithm, we show the average gap to the optimal results and present the 95% confidence interval analysis as a function of the protected tasks percentage. We can see that quality of both heuristics grows with the increase of the protected tasks percentage. Summarizing all tests, results of AlgGreedy were on average 10.19% worse than the optimal ones. The corresponding value for AlgRand was 19.73%.

Table 1. Average performance of heuristic algorithms in terms of the OPEX cost compared with the optimal results as a function of the protected tasks percentage; 30 computing nodes

Protected tasks percentage (PTP)	AlgGreedy		AlgRand	
	Gap to the optimal results	Lengths of 95% confidence intervals	Gap to the optimal results	Lengths of 95% confidence intervals
0%	12.42%	0.72%	23.52%	1.87%
10%	11.64%	0.83%	22.72%	1.79%
20%	11.04%	0.76%	21.87%	1.69%
30%	10.62%	0.72%	20.76%	1.86%
40%	10.32%	0.57%	20.17%	1.96%
50%	9.76%	0.61%	19.26%	1.94%
60%	9.47%	0.63%	18.32%	1.68%
70%	9.16%	0.60%	17.60%	1.60%
80%	8.81%	0.69%	16.70%	1.58%
90%	8.69%	0.65%	16.54%	1.38%
100%	8.41%	0.72%	15.71%	1.24%

To verify the scalability of the proposed heuristic approach, larger computing systems including 100 nodes and 60 projects were generated at random in a similar way as above. Table 2 presents results obtained for these systems. We report performance of the AlgGreedy algorithm compared to CPLEX. However, in this case CPLEX did not provide optimal results, since after about 3 minutes, the solver was stopping calculation with the out-of-memory error. Nevertheless, CPLEX returned the current solution (non-optimal), as well as the current value of the lower bound. Recall that the searched optimal result is located between these two values. In Table 2, we compare AlgGreedy against the lower bound and CPLEX results. Heuristic results were 14.02% and 12.53 worse on average (taking into account all 396 cases) than the lower bounds and CPLEX results, respectively. It can be noticed that efficiency of AlgGreedy for 100-node systems was slightly lower than for 30-node system evaluated in Table 1. This follows mainly from the fact that the solution space of larger systems (e.g., 100-node) is significantly larger compared to smaller 30-node systems. AlgGreedy is also a relatively simple heuristic that looks for the solution is a single run of the algorithm. However, results of AlgGreedy are still on a satisfactory level.

We also compared AlgGreedy against AlgRand, and the gap between both algorithms was on a similar level as in the case of 30-node systems. It should be noted that for systems larger than 100 nodes and 60 projects, CPLEX usually could not provide any feasible results or lower bounds due to huge memory requirements. Therefore, only heuristic algorithms can be applied in such cases.

Table 2. Average performance of the AlgGreedy algorithm in terms of the OPEX cost compared with the lower bound and results of CPLEX as a function of the protected tasks percentage obtained for 100-node and 60-project systems

Protected tasks percentage (PTP)	AlgGreedy		AlgGreedy	
	Gap to the lower bound	Lengths of 95% confidence intervals	Gap to the CPLEX results	Lengths of 95% confidence intervals
0%	14.32%	0.45%	12.85%	0.39%
10%	14.23%	0.44%	12.78%	0.41%
20%	13.28%	0.52%	11.91%	0.52%
30%	14.00%	0.47%	12.61%	0.40%
40%	13.71%	0.50%	12.13%	0.48%
50%	13.88%	0.58%	12.41%	0.55%
60%	14.42%	0.56%	12.91%	0.51%
70%	13.97%	0.52%	12.52%	0.46%
80%	13.77%	0.44%	12.17%	0.44%
90%	13.94%	0.56%	12.36%	0.51%
100%	14.74%	0.58%	13.18%	0.54%

The next goal was to investigate how additional survivability constraints expressed by the protected tasks percentage parameter influence the system cost. In addition to 30-node networks, we created larger problem instances including 200 computing nodes and 1200-4800 tasks. Other parameters were analogous to the case of 30-node systems. Only the number of destination nodes was different: in range 1-8, and transmission rate of input and output data was in range 5-10Mbps. For these larger systems, CPLEX also could not return solutions due to the out-of-memory error. Therefore, for 200-node networks, only results of AlgGreedy are presented here.

In Figs. 4÷5 we show the average values of the OPEX cost for 30-node and 200-node systems, respectively. Note that in the case of 30-node systems, Fig. 4 reports results of CPLEX, while in the case of 200-node systems (Fig. 5), results are obtained by AlgGreedy. Each figure shows costs related to link capacity and processing. To present the aggregate results, for each case (unique in terms of the computing system and task set), we calculated the relative cost (capacity and processing) normalized using the cost obtained for PTP = 0%. Next, the average value for each PTP was computed. The trend in both figures is similar, i.e., both kinds of cost grow linearly with the increase of the PTP parameter. However, processing cost grows in a more dynamic way compared to the capacity cost. It follows from the fact that relatively cheaper nodes (i.e., with lower values of capacity and processing costs) are saturated prior to more expensive nodes.

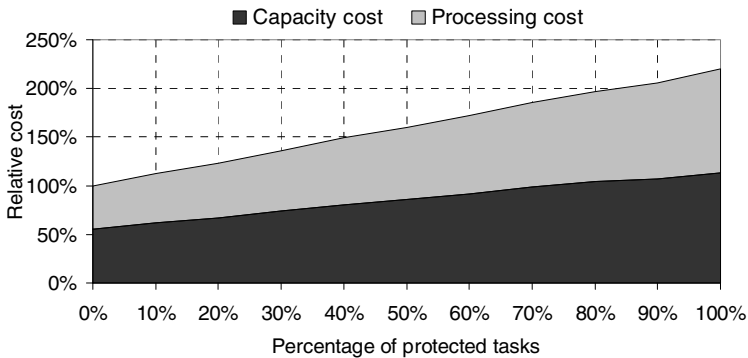


Fig. 4. Average relative cost as a function of protected tasks percentage for 30-node networks

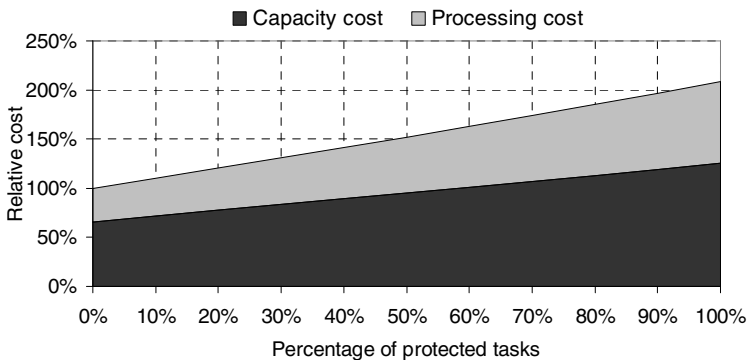


Fig. 5. Average relative cost as a function of protected tasks percentage for 200-node networks

Therefore, when the number of tasks to be protected grows, some tasks have to be allocated to more costly nodes. Moreover, we can notice that the average cost of systems with full protection (PTP = 100%) is about 110% and 106% higher, compared to unprotected tasks (PTP = 0%) for 30-node, and 200-node systems, respectively.

7 Conclusion

In this paper, we focused on 1+1 dedicated protection of overlay distributed computing systems against failures of single nodes. A new ILP model of joint optimization of task allocation and link capacity assignment has been proposed. Efficient heuristic algorithms have been introduced to find solutions for large problem instances. Our approach thoroughly addresses the issue of minimizing the operational (OPEX) cost of distributed computing systems (i.e., including processing costs and network capacity costs) designed to provide protection against node failures. Extensive numerical experiments proved the efficiency of the proposed heuristic algorithms.

In future work we plan to incorporate the concept of shared protection into our model (i.e., sharing the backup resources) to reduce the cost of protection provisioning.

Acknowledgement. This work was supported in part by the National Science Centre (NCN), Poland.

References

1. Travostino, A., Mambretti, J., Karmous-Edwards, G. (eds.): *Grid Networks Enabling Grids with Advanced Communication Technology*. Wiley (2006)
2. Nabrzyski, J., Schopf, J., Węglarz, J. (eds.): *Grid Resource Management: State of the Art and Future Trends*. Kluwer Academic Publishers (2004)
3. Milošević, D., et al.: *Peer to Peer computing*. HP Laboratories Palo Alto, HPL-2002-57 (2002)
4. Wilkinson, B.: *Grid Computing: Techniques and Applications*. Chapman & Hall/CRC Computational Science (2009)
5. Grover, W.D.: *Mesh-Based Survivable Networks: Options and Strategies for Optical, MPLS, SONET, and ATM Networking*. Prentice Hall PTR, New Jersey (2003)
6. Ramamurthy, S., et al.: *Survivable WDM Mesh Networks*. *IEEE/OSA Journal of Lightwave Technology* 21(4), 870–883 (2003)
7. Pióro, M., Medhi, D.: *Routing, Flow and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann (2004)
8. Molisz, W., Rak, J.: *Region Protection/Restoration Scheme in Survivable Networks*. In: Gorodetsky, V., Kotenko, I., Skormin, V.A. (eds.) *MMM-ACNS 2005*. LNCS, vol. 3685, pp. 442–447. Springer, Heidelberg (2005)
9. Ramamurthy, S., Mukherjee, B.: *Survivable WDM Mesh Networks, Part I – Protection*. In: *Proc. IEEE INFOCOM 1999*, vol. 17(2), pp. 43–48 (1999)

10. Develder, C., et al.: Survivable Optical Grid Dimensioning: Anycast Routing with Server and Network Failure Protection. In: Proc. of IEEE ICC 2011, pp. 1–5 (2011)
11. Thysebaert, P., et al.: Scalable Dimensioning of Resilient Lambda Grids. *Future Generation Computer Systems* 24(6), 549–560 (2008)
12. Buysse, J., De Leenheer, M., Dhoedt, B., Develder, C.: Providing Resiliency for Optical Grids by Exploiting Relocation: A Dimensioning Study Based on ILP. *Computer Communications* 34(12), 1389–1398 (2011)
13. Jaumard, B., Shaikh, A.: Maximizing Access to IT Services on Resilient Optical Grids. In: Proc. of 3rd International Workshop on Reliable Networks Design and Modeling, RNDM 2011, pp. 151–156 (2011)
14. Rak, J.: Fast Service Recovery under Shared Protection in WDM Networks. *IEEE/OSA Journal of Lightwave Technology* 30(1), 84–95 (2012)
15. Vasseur, J.P., Pickavet, M., Demeester, P.: *Network Recovery*. Elsevier (2004)
16. Luo, H., Li, L., Yu, H.: Insights for Segment Protection in Survivable WDM Mesh Networks with SRLG Constraints. In: Proc. IEEE GLOBECOM 2008, pp. 1–5 (2006)
17. Tapolcai, J., Ho, P.-H., Verchere, D., Cinkler, T., Haque, A.: A New Shared Segment Protection Method for Survivable Networks with Guaranteed Recovery Time. *IEEE Transactions on Reliability* 57(2), 272–282 (2008)
18. Rak, J.: Capacity Efficient Shared Protection and Fast Restoration Scheme in Self-Configured Optical Networks. In: Keller, A., Martin-Flatin, J.-P. (eds.) *SelfMan 2006*. LNCS, vol. 3996, pp. 142–156. Springer, Heidelberg (2006)
19. Song, L., Mukherjee, B.: Accumulated-Downtime-Oriented Restoration Strategy With Service Differentiation in Survivable WDM Mesh Networks. *IEEE/OSA Journal of Optical Communications and Networking* 1(1), 113–124 (2009)
20. Guo, L., Li, L.: A Novel Survivable Routing Algorithm With Partial Shared-Risk Link Groups (SRLG)-Disjoint Protection Based on Differentiated Reliability Constraints in WDM Optical Mesh Networks. *IEEE/OSA Journal of Lightwave Technology* 25(6), 1410–1415 (2007)
21. Kacprzak, T., Walkowiak, K., Woźniak, M.: Optimization of Overlay Distributed Computing Systems for Multiple Classifier System – Heuristic Approach. *Logic Jnl IGPL* (2011), doi:10.1093/jigpal/jzr020
22. Shen, X., Yu, H., Buford, J., Akon, M. (eds.): *Handbook of Peer-to-Peer Networking*. Springer (2009)
23. Anderson, D.: BOINC: A System for Public-Resource Computing and Storage. In: Proc. of the Fifth IEEE/ACM International Workshop on Grid Computing, pp. 4–10 (2004)
24. Akbari, B., Rabiee, H.R., Ghanbari, M.: An Optimal Discrete Rate Allocation for Overlay Video Multicasting. *Computer Communications* 31(3), 551–562 (2008)
25. Zhu, Y., Li, B.: Overlay Networks with Linear Capacity Constraints. *IEEE Transactions on Parallel and Distributed Systems* 19(2), 159–173 (2008)
26. ILOG AMPL/CPLEX software, <http://www.ilog.com/products/cplex/>

Scheduling and Capacity Design in Overlay Computing Systems

Krzysztof Walkowiak, Andrzej Kasprzak, Michał Kosowski, and Marek Miziołek

Department of Systems and Computer Networks, Faculty of Electronics, Wrocław
Wrocław University of Technology, Wybrzeże Wyspińskiego 27, 50-370 Wrocław, Poland
Krzysztof.Walkowiak@pwr.wroc.pl

Abstract. Parallel to new developments in the fields of computer networks and high performance computing, effective distributed systems have emerged to answer the growing demand to process huge amounts of data. Comparing to traditional network systems aimed mostly to send data, distributed computing systems are also focused on data processing what introduces additionally requirements in the system performance and operation. In this paper we assume that the distributed system works in an overlay mode, which enables fast, cost-effective and flexible deployment comparing to traditional network model. The objective of the design problem is to optimize task scheduling and network capacity in order to minimize the operational cost and to realize all computational projects assigned to the system. The optimization problem is formulated in the form of an ILP (Integer Linear Programming) model. Due to the problem complexity, four heuristics are proposed including evolutionary algorithms and Tabu Search algorithm. All methods are evaluated in comparison to optimal results yielded by the CPLEX solver. The best performance is obtained for the Tabu Search method that provides average results only 0.38% worse than optimal ones. Moreover, for larger problem instances with 20-minute limit of the execution time, the Tabu Search algorithm outperforms CPLEX for some cases.

Keywords: distributed computing, overlays, ILP modeling, optimization.

1 Introduction

Recent advances in networking and distributed computing research have enabled expansion of effective distributed systems including Grid computing systems to support high-performance applications in many areas of human activity, e.g., medical data analysis, climate/weather modeling, bioinformatics, experimental data acquisition, collaborative visualization of large scientific databases, financial modeling, earthquake simulation, astrophysics and many others [1]-[8]. Deployment of distributed computing systems requires many studies embracing a wide range of various research challenges. In this work we focus of one of these aspects and address the problem of scheduling and capacity design in overlay computing systems.

The distributed computing systems like Grids can be developed using special dedicated high-speed networks [1] as well as the Internet can be used as the backbone

network for an overlay-based systems [6]-[8]. In this paper we focus on the second case, since overlay systems provide considerable network functionalities (e.g., diversity, flexibility, manageability) in a relatively simple and cost-effective way as well as regardless of physical and logical structure of underlying networks. The considered system consists of a set of computing elements spread geographically (e.g., clusters or other hardware equipment). The workflow of the system assumes that input data generated in one of the nodes is transmitted to a computing nodes that is responsible to process the data and finally to deliver the results to all destination nodes that are interested in results of computations. Such a workflow modeling is generic and fits to numerous applications aimed to be applied in distributed manner, e.g., image processing, data analysis, numerical computations. The objective of the optimization is to minimize the operational cost (OPEX) of the computing system including expenses related to two most important resources, i.e., processing equipment and network connections.

The main novelty of this work – comparing to previous papers on the topic of Grid optimization – is joint optimization of scheduling and network capacity in overlay-based systems, while most of former papers considered either optical networks (e.g., [9]-[11] or only scheduling optimization (e.g., [12]). In this paper we continue our research on optimization of distributed systems. Note that in our previous work [12] we studied an optimization problem similar to the model addressed in this paper. The major modification of the current work is joint optimization of task scheduling and network capacity, while in [12] only the task allocation was analyzed, since network capacity was given.

The main contributions of the paper are threefold. (i) A new ILP model of scheduling and capacity design in overlay computing system formulated in accordance with assumptions following from real overlay and computing systems. (ii) Four heuristic algorithms proposed to solve the ILP model. (iii) Results of extensive numerical experiments run to adjust tuning parameters of heuristics as well as to examine the performance of the overlay computing system in various conditions.

The rest of the paper is organized as follows. Section 2 discusses the works related to optimization of networks and distributed computing systems. In Section 3, we present the system architecture and formulate the scheduling and capacity design in overlay computing systems as the ILP model. Section 4 introduces four heuristic algorithms proposed to solve the addressed problem. In Section 5, results of numerical experiments are reported. Finally, the last section concludes the work.

2 Related Works

The authors of [9] introduce several multicost algorithms for a joint scheduling of the communication and computation resources. They present a multi-cost scheme of polynomial complexity that provides reservations and selects the computation resource to execute the task and determines the path to route the input data. The result of numerical experiments shows that in a Grid network in which tasks are either CPU- or data-intensive (or both), it is beneficial for the scheduling algorithm to jointly consider the computational and communication problems.

The paper [10] addresses the optimization of optical Grids considering combined dimensioning and workload scheduling problem with additional survivability requirements. The Divisible Load Theory is applied to tackle the scalability problem and compare non-resilient lambda Grid dimensioning to the dimensions needed to survive single-resource failures. For regular network topologies, analytical bounds on the dimensioning cost are obtained. To validate these bounds, the authors report results of comparisons for the resulting Grid dimensions assuming a 2-tier Grid operation as a function of varying wavelength granularity, fiber/wavelength cost models, traffic demand asymmetry and Grid scheduling strategy for a specific set of optical transport networks.

The authors of [11] focus on dimensioning problem of computing Grids taking into account server location and capacity, network routing and capacity. They propose an integrated solution with joint optimization of the network and server capacity, and incorporate resiliency against both network and server failures. A case study on a meshed European network comprising 28 nodes and 41 links is presented. The results show that compared to classical (i.e., without relocation), they can offer resilience against both single link and network failures by adding about 55% extra server capacity, and 26% extra wavelengths.

In [12], the authors consider optimization of overlay computing systems. The main goal is to allocate tasks to computing nodes in order to minimize the operational cost related to data transmission and processing. An ILP model is formulated and effective heuristic based on the GRASP method is proposed and evaluated. Note that the problem considered in this paper uses similar assumptions as [12]. The main novelty is that additionally network capacity dimensioning is applied in this work.

For more information related to Grid systems including description of representative examples of Grid computing projects refer to [1]-[5]. More information on overlay systems can be found in [6]-[8].

3 Modeling of Scheduling and Capacity Design in Overlay Computing System

The optimization model of overlay computing system is formulated consistent with characteristics of real overlay systems as well as according to assumptions presented in previous works [6]-[14]. The model is generic as various computing systems, e.g., Grids, public resource computing systems fit to the model [1]-[5].

3.1 Notation

The computing system consists of nodes indexed by $v = 1, 2, \dots, V$. The nodes represent computing elements (individual computers or clusters) as well as sources of input data and destinations of output data. The system works on the top of an overlay network (e.g., Internet) and each node is connected by an access link to the network. The connectivity between nodes is provided by virtual links of the overlay realized by paths consisting of links realized in the underlying network. The main motivation to use the

overlay idea in computing systems is large flexibility – many currently deployed network systems apply the overlay approach, e.g. Skype, IPTV, Video on Demand, SETI@home, etc. According to [14], nodes' capacity constraints are typically adequate in overlay networks. Additionally, in overlays typically the underlay physical network is assumed to be overprovisioned and the only bottlenecks are access links [13]. Therefore, the only network capacity constraints in the model refer to access links. Since the access link capacity is to be dimensioned, integer variable y_v denotes the number of capacity modules allocated to the access link of node v . We assume that each node v is assigned to a particular ISP (Internet Service Provider), which offers high speed access link with a capacity module m_v given in Mbps (e.g., Fast Ethernet). We assume that each node is already equipped with some computers and p_v denotes the maximum processing of node v given by a number of uniform computational tasks that node v can calculate in one second. Nevertheless, the problem can be easily modified to allow dimensioning of node's processing power by introducing additional integer variable.

The computing system is to process a set of computational projects indexed by $r = 1, 2, \dots, R$. Each project consists of uniform computational tasks (of the same required processing power, e.g., a number of FLOPS) – the number of tasks in project r is given by n_r . Each task of the project can be processed independently, i.e., we assume that there is no dependency between individual tasks. Each project is assigned with a source node that produces the input data and one or more destination nodes that are to receive the output data including results of computations (processing). Constant $s(r, v)$ is 1, if node v is the source node of project r ; 0 otherwise. In the same way, $t(r, v)$ is 1, if node v is the destination node of project r ; 0 otherwise. In spite of the assumptions that the uniform task for each project has the same computational requirement, the values of the input and output data transmit rates are specific for each computational project following from particular features of the project. Thus, constants a_r and b_r denote the transmit rate of input data and output data, respectively, per one task in project r and are given in bps (bits per second).

The workflow of the system is as follows. The input data of the project is transferred from the source node providing input data to one or more computing nodes that processes the data. Next, the output data (results of computations) is sent from each computing node to one or more destination nodes. We assume similar to [10]-[12] that computational projects are long-lived, i.e., they are established for a relatively long time (e.g., days, weeks). As an example of long-lived project we can enumerate scientific experiments like LHC (Large Hadron Collider). The input and output data associated with the project is continuously generated and transmitted. Hence, computational and network resources can be allocated in the system according to an offline optimization. To keep the model relatively simple and enable optimization of large problem instances, we do not consider – as in many other works (e.g., [1], [2], [9] – the time dependency of each task (starting time, completion time, etc.). However, the model can be modified to incorporate additional constraints. An integer variable x_{rv} denotes the number of project r tasks allocated to node v . A corresponding auxiliary binary variable z_{rv} is 1 if node v calculates at least one task of project r , 0 otherwise.

We assume that the maximum number of computing nodes involved in one project cannot be larger than N . For instance, if $N = 1$, then all uniform tasks of a particular project can be computed only on one node. When $N = V$, the number of computing nodes is not limited. We refer to N as *split ratio*. The motivation behind this parameter follows from management issues, i.e., less computing node (lower value of the split ratio) facilitates the management of the computing system.

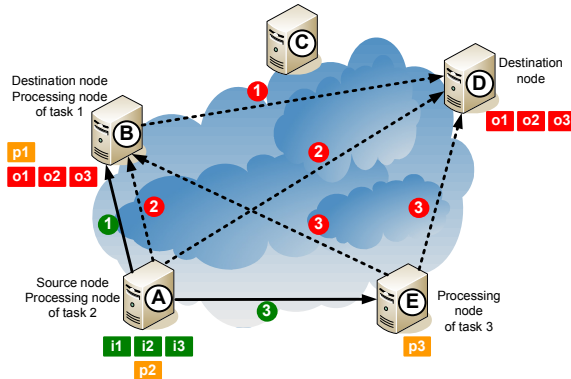


Fig. 1. Example of a distributed computing system

Fig. 1 presents a simple example to illustrate the computing system architecture. The system contains five computing nodes denoted as A, B, C, D, and E. Three tasks are to be computed. Node A is the source node of all tasks. Therefore, it stores input data related to the tasks (green rectangles labeled i_1 , i_2 and i_3). Orange rectangles labeled p_1 , p_2 , and p_3 denote the places where a particular task is calculated. Red rectangles labeled o_1 , o_2 , and o_3 denote results of computations related to tasks 1, 2 and 3, respectively. Nodes B and D are destinations of all tasks. Additionally, we show network flows generated to deliver the input and output data. Solid line denotes the flow of input data. Green circle with a number inside shows the indices of tasks the input data is related to. Dotted line shows the flow of output data. Again, the numbers in red circles indicate task indices the data belongs to.

3.2 Objective Function

The objective of the problem is to minimize the operating cost (OPEX) of the computing system including expenses related to two elements: network and processing. Since the overlay network is used, from the perspective of the computing system the only network costs are related to the access link, i.e., we do not include in the model costs related to the backbone network covered by network operators. Constant ξ_v , given in euro/month denotes the whole OPEX cost related to one capacity module allocated for node v and includes leasing cost of the capacity module paid to the ISP as well as all other OPEX costs like energy, maintenance, administration, etc. As computing nodes are spread geographically in many countries with various ISPs, energy and maintenance costs, values of the module size and cost can be different for each

node. Note that – if necessary – a number (more than 1) of capacity modules can be assigned to node v – thus the decision variable y_v is integer. For a good survey on various issues related to network costs see [15].

Constant ψ_v denotes the OPEX cost related to processing of one uniform task in node v . The ψ_v cost is defined in euro/month and contains all expenses necessary to process the uniform computational tasks including both processing and storage issues (e.g., energy, maintenance, administration, hardware amortization etc.). Similarly to the network cost given by ξ_v – processing cost ψ_v is different for various nodes. This follows from the fact that nodes are placed in different countries with various costs related to energy, maintenance, administration, etc. Various aspects of grid economics are discussed in [1], [2].

3.3 ILP Model

indices

$v, w = 1, 2, \dots, V$	computing nodes
$r = 1, 2, \dots, R$	projects

constants

p_v	maximum processing rate of node v
n_r	size of project r (number of tasks in project)
a_r	transmit rate of input data per one task in project r (Mbps)
b_r	transmit rate of output data per one task in project r (Mbps)
$s(r, v)$	= 1 if v is the source node of project r ; 0 otherwise
$t(r, v)$	= 1 if v is the destination node of project r ; 0 otherwise
t_r	number of destination nodes in project r
M	large number
N	split ratio
ψ_v	OPEX cost related to processing of one task in node v (euro/month)
ξ_v	OPEX cost related to one capacity module of node v (euro/month)
m_v	size of the capacity module for node v (Mbps)

variables

x_{rv}	number of tasks of project r assigned to node v (integer)
y_v	capacity of access link for node v expressed in the number of modules (non-negative integer)
z_{rv}	= 1, if project r is calculated on node v ; 0, otherwise (binary)

objective

$$\min C = \sum_v y_v \xi_v + \sum_v \sum_r x_{rv} \psi_v \tag{1}$$

constraints

$$\sum_r x_{rv} \leq p_v \quad v = 1, 2, \dots, V \tag{2}$$

$$\sum_r (1 - s(r, v)) a_r x_{rv} + \sum_{r: t(r, v)=1} b_r (n_r - x_{rv}) \leq y_v m_v \quad v = 1, 2, \dots, V \tag{3}$$

$$\sum_{r: s(r, v)=1} a_r (n_r - x_{rv}) + \sum_r (t_r - t(r, v)) b_r x_{rv} \leq y_v m_v \quad v = 1, 2, \dots, V \tag{4}$$

$$\sum_v x_{rv} = n_r \quad r = 1, 2, \dots, R \quad (5)$$

$$x_{rv} \leq Mz_{rv} \quad r = 1, 2, \dots, R \quad v = 1, 2, \dots, V \quad (6)$$

$$\sum_v z_{rv} \leq N \quad r = 1, 2, \dots, R \quad (7)$$

The objective (1) is the OPEX cost related to access links selected for each node and processing costs of computational tasks. Constraint (2) is in the model to guarantee that the number of tasks assigned to each node cannot be larger than the node's processing limit given by p_v . Condition (3) assures the download capacity constraint for each node. The left-hand side of (3) defines the flow entering node v including two terms. The former one (i.e., $\sum_r (1 - s(r,v))a_r x_{rv}$) is the transmit rate of input data of all projects calculated on node v . If the node v is the source node of project r (i.e., $s(r,v) = 1$), then this flow is 0. The latter term (i.e., $\sum_{r:t(r,v)=1} b_r (n_r - x_{rv})$) follows from the fact that each destination node v of the project r (i.e. $t(r,v) = 1$) must download the project results related to $(n_r - x_{rv})$ tasks (all tasks of project r except the tasks assigned to the node v). The right-hand side of (3) denotes the download capacity of node v . Similar to (3), constraint (4) defines the upload capacity constraint. Again, the left-hand side comprises two elements. The first one (i.e., $\sum_{r:s(r,v)=1} a_r (n_r - x_{rv})$) assures that the source node v of project r ($s(r,v) = 1$) must upload input data for computation of $(n_r - x_{rv})$ tasks (all tasks excluding the tasks assigned to node v). The second element (i.e., $\sum_r (t_r - t(r,v))b_r x_{rv}$) means that each node must upload the output data to all destination nodes of the project r given by t_r . Notice that we take into account the case when node v is among destination nodes of the project r , therefore we use formula $(t_r - t(r,v))$ to define the number of nodes to which the output data is transmitted. To assure that for each project $r = 1, 2, \dots, R$ all task are assigned for processing we formulate constraint (5). The model given by (1)-(5) defines the basic version of the scheduling and capacity design problem for overlay computing systems. Additionally, we define a limit on the maximum number of nodes involved in each project. The idea is to minimize the management overhead and reduce the number of nodes processing tasks related to the same project. Therefore, we introduce constraint (6) that binds the binary variable z_{rv} with decision variable x_{rv} . Condition (7) is in the model to meet the requirement that for each project r , the number of nodes involved in the project cannot exceed the given split ratio N .

4 Heuristic Algorithms

The ILP model given by (1)-(7) is NP-complete (it is equivalent to network design problem with modular links [15]). Therefore, to solve the model in optimal way, exact methods including branch-and-bound or branch-and-cut must be applied. However, only relatively small problem instance can be solved in optimal way in reasonable time. Consequently, to solve the problem for larger instance, effective heuristic are required. In this section, we present four heuristics: random algorithm (RA), greedy algorithm (GA), evolutionary algorithm (EA) and Tabu Search algorithm (TA).

4.1 Random Algorithm

The random method follows from common features of existing Peer-to-Peer overlay systems, which mostly operate in a random manner, e.g., BitTorrent [6]-[8]. First, the RA algorithm selects at random a task, which is not yet allocated to a computing node. Let's assume that the task is of project r . Next, the task is assigned to a node according to the following procedure. If the split ratio limit is not achieved (i.e., the number of nodes involved in the considered project r is lower than N), a node with free resources of processing power (constraint (2)) is selected at random. Otherwise, when the current split ratio of project r is N , one of nodes already involved in the current project and with spare processing resources is chosen at random. When the processing node is selected, the considered task is allocated to this node. In this way values of variables x_{rv} are assigned. Note that when there are no feasible nodes (i.e., all possible nodes for the selection already meet the processing limit), the algorithm returns information that no feasible solution exists. Finally, to find values of capacity variables y_v , all nodes are assigned with a sufficient number of capacity modules to satisfy network flows following from assignment of tasks. Execution of the algorithm is repeated 20 times, and the final result is the minimum value of the obtained cost.

4.2 Greedy Algorithm

The next algorithm is based on the greedy idea. All tasks are processed in a single run of the algorithm. As in the case of the random approach, only feasible nodes (with free processing resources – constraint (2)) are considered as candidate nodes. In each iteration of the algorithm, the cheapest allocation of one task of project r to node v is made according to metric c_{rv} being the weighted sum of two elements (see below). In more details, for all not yet allocated projects (i.e., with at least one not allocated task), the current value of metric c_{rv} is calculated. To satisfy the split ratio constraint (7), when the current number of nodes involved in the project r reached the split ratio N , only nodes incorporated in project r are taken into account in calculation of c_{rv} . Next, one task of the selected project is assigned to the node v , guaranteeing the lowest value of c_{rv} .

Now we describe the procedure how the GA algorithm calculates the c_{rv} metric. The first component of c_{rv} is calculated as an average cost of allocation of one task of project r to node v taking into account the processing cost as well as the network cost. The network cost follows from the fact that allocation of the task to node v generates (i) flow of input data from the source node of the task to node v and (ii) flow of the output data from node v to all destination nodes of the task. For all these flows, network capacity is required. For instance, the cost related to the download capacity of the computing node v and one task of project r is estimated by formula $\xi_v a_r / m_v$. All other costs are calculated in analogous way. The second element of c_{rv} approximates the additional costs (both processing and network) that would be required in the current system to serve allocation of a new task of project r to node v . The additional processing cost is simply given by ψ_v . The additional network cost would be necessary only when the allocation of one task of project r to node v generates the need to add a new capacity module(s) to some nodes (to provide enough capacity to send input/output data). Note that allocation of the new task to node v may not require

augmenting of existing capacity, since the already allocated resources may be sufficient to send all the required data.

4.3 Evolutionary Algorithm

To solve the optimization problem formulated in Section 3, a new evolutionary algorithm [16], [17] was proposed. Both operators and the scheme were adjusted to the given problem. First, we introduce the chromosome coding. We assume that the chromosome is represented as a two dimensional matrix with V columns and R rows denoting variables x_{rv} (i.e., how many tasks of each project are assigned to each node). Such a problem representation may seem too complex for the evolutionary algorithm, but it was applied in the past with success in the context of both transportation and state assignment problems. This is the most natural representation and also gives the opportunity to modify represented individuals in many ways. In the case of a sparse matrix (e.g., when the split ratio has a low value), an additional data structure holding the position of nonzero values is created to increase the algorithm's effectiveness.

Now we introduce operators of the algorithm. Two tuning parameters are used in the mutation operator: *mutation_probability_1* and *mutation_probability_2*. The former parameter denotes the probability of an individual selection to perform the mutation operation. The latter parameter is the probability of a particular row selection of a given individual matrix. When row r is selected to run the mutation operator, the following procedure is applied. Two elements of the row are selected, but one of the selected elements must have value greater than 0, while the second elements must be equal to 0. Next, values between these two elements from the given row are swapped. However, it happens only if such a change does not affect constraints of the problem.

The crossover operator is much more complex. It creates one offspring with the use of two parents. It also works sequentially for each row of the matrix. Operations are made for each row as long as a right number of tasks is assigned to the project in offspring individual. Every time a pair of genes, one from each parent is chosen. Gene in that case means a particular variable x_{rv} (i.e., a number of tasks of project r assigned to node v). The selection of the parent genes is based on one of three criteria: (i) cost of processing on a particular node v (i.e., ψ_v), (ii) cost of a link capacity module of node v (i.e., ξ_v), (iii) node v is a source or a destination node of project r what automatically reduces the amount of data that is to be sent in the system. A final criterion is chosen randomly and it decides, which gene is chosen for the offspring individual. The number of tasks hold by the gene can be reduced because of existing constraints. Operations are repeated until a valid offspring satisfying all constraints is created.

The initial population of the algorithm is created at random, however all individuals must be valid. The base population includes *population_size* individuals. Then, as long as it is specified according to the time limit, the following operations are executed. At the beginning of the loop, the mutation operator is run. The next operation is selection. In the proposed algorithm the ranking selection is applied. Comparing to other methods (i.e., tournament and roulette), this approach yielded the best results according to initial tests. It selects the parents that will be used to create the offspring. Then, the crossover operator is used. It is done in a loop executed *offspring_population_size* times, so the same number of offsprings is created. Later,

both new and old individuals are used to form the population for the next iteration. The algorithm chooses the best individuals (those with the lowest overall cost) which meet the age limitation. Each individual has its age, which is increased at the end of the algorithm's iteration. In the proposed algorithm the highest possible age is denoted by parameter *individual_age*. The formed population for the next generation has *population_size* valid individuals.

The EA algorithm includes additional operators to cope with the split ratio constraint (7). The operators increase or decrease the split ratio of the individual. The former operator is executed according to the *increase_split_frequency* parameter, for instance, if *increase_split_frequency* is 10 then this operator is run at 10th, 20th, etc. population. The execution of the latter operator is triggered consistent with the *decrease_split_frequency* parameter. The split ratio decrease operator is run for each project r . First, among all nodes that participate in project r (i.e., with $x_{rv} > 0$) the most expensive node (in terms of the processing and networking costs) is selected. Next, all tasks assigned to this node are tried to be reallocated to other nodes in order to decrement the number of nodes assigned to project r . However, the processing limit constraint (2) cannot be violated. The split ratio increase operator is applied only to projects for which the number of nodes involved in the current project is lower than N . For such a project, the cheapest node (in terms of the processing and networking costs) not involved in project is identified. Next, tasks from other nodes are reallocated to this node. However, the number of new tasks assigned to this node cannot exceed its processing limit and one third of the number of tasks in the given project.

4.4 Tabu Search Algorithm

The general scheme of the proposed TA algorithm follows from classical Tabu Search algorithm [18]. Below we describe the basic operations of the algorithm with a special focus on new elements. The starting solution is generated at random using a following procedure in order to create a feasible solution in terms of the problem constraints – especially the most challenging is the split ratio constraint (7). Projects are processed sequentially according to decreasing values of n_r (number of tasks in the project). For each project, the following steps are made. First, computing nodes are sorted in descending order by the current residual processing capacity (i.e., the difference between p_v and the number of task already allocated to node v). Next, tasks of analyzed project r are randomly assigned to N first nodes from the list of previously sorted nodes – in this way the split ratio constraint (7) is satisfied. If in a particular case, a feasible solution is not generated (i.e., some constraints are violated), the procedure for the considered project is repeated. It should be noted that the algorithm as a starting solution can use also any feasible solution yielded by another algorithm.

The next important operation of the TA algorithm are *moves*, which describe the modifications applied to solutions during the neighborhood exploration. Solution of discussed problem consists of two parts: matrix x_{rv} and vector y_v . The former element defines how many tasks from project r are assigned for processing in node v , the latter element holds the information about the number of capacity modules assigned to each node. Any transformation of such a solution consists of moving some number of tasks

(e.g., t tasks) of project r from node v to node w and recalculation of variables y_v and y_w accordingly. For easier notation, we define a *swap* operator $S[r, v, w, t]$ denoting that t tasks of project r are moved from node v to node w .

The subsequent element of the algorithm is to generate a neighborhood of size n from the current solution to search for a new, better solution. The neighborhood is generated by taking the current solution and transforming it n times using the swap operation. Each transformation is unique and is always applied to the current solution. As a result, we get a list of n solutions created from the current one. Then, the cost of every neighborhood solution is evaluated and the best one is chosen. Every better solution in the neighborhood is checked to fulfill the tabu list and aspiration rule requirements. In the end, the neighborhood solution with the lowest cost (which can be worse than the current one) is chosen as the new current one. All executed transformations, described as a set of swap operations, are used to update the tabu list.

Two variants of the neighborhood exploration are proposed: exploration by node and exploration by project. The former method starts with a random selection of a node denoted by v . Next, for each project r that is assigned to node v (i.e., $x_{rv} > 0$) the following procedure is executed. Tasks of the project are tried to be reallocated to another node – all remaining nodes are analyzed. However, the obtained solution must be feasible, i.e., the processing limit constraint (2) and the split ratio constraint (7) must be satisfied. The number of start nodes is defined by parameter called *exploration_range*. To limit the search time of big neighborhoods, only *exploration_deep* moves for each start node are tested. In the exploration by project approach, the algorithm starts from a randomly selected start project r . Then, all nodes which are allocated with tasks of this project are processed. For each of those nodes, an attempt is made to move all tasks belonging to project r , from the current node to another node taking into account all remaining nodes. As in the previous case, only moves yielding feasible solutions in terms of constraint (2) and (7) are considered. Again, parameters *exploration_range* and *exploration_deep* can be applied to tune the procedure.

The main purpose of the tabu list is to avoid the algorithm from falling into cycles during the search. It is the key element of the Tabu Search algorithm. There are many theoretical divagations and practical examples of different tabu list constructions [18]-[20]. The tabu list in the discussed algorithm maintains a list of last t swaps, which were used for creating the current solution. For the purpose of our algorithm, two blocking rules were developed with a possibility to use any combination of them:

- Equality rule – each swap operation included in the tabu forbids any move that contains the same swap.
- Similarity rule – each swap operation included in the tabu forbids any move (swap) that contains the same to/from node and project, i.e., also swaps with a different number of moved tasks than the one included in the tabu are forbidden.

The aspiration rule is implemented according to [17]. If a new solution is blocked by the tabu list, the aspiration rule checks if its cost is lower or equal comparing to the cost of the best solution already been found. If this condition is met, the tabu blockade is ignored and a new solution is accepted as a current one and as the best one. The algorithm's stop condition is the execution time.

5 Results

In this section, we present the results of computational experiments. All algorithms described in the previous section were implemented using C++ . Moreover, the ILP model presented in Section 3 was solved to obtain optimal results using a branch-and-cut algorithm included in CPLEX 11.0 solver [21]. The goal of experiments was threefold. First, we wanted to examine the performance of heuristics according to tuning parameters of the algorithms. Second, we compared results of heuristic methods against optimal results provided by CPLEX. Finally, we analysed how the split ratio parameter influences the OPEX cost of the distributed computing system. Simulations were run on a PC computer with IntelCore i7 processor and 4GB RAM.

Various distributed computing systems were generated at random with the following two sizes: small systems including 20 nodes and 10 projects and large systems including 200 nodes and 120 projects. Table 1 contains detailed information about ranges of system parameters applied in the random generation of the systems. For each size of the system, 162 unique configurations were tested, i.e., 27 configurations of nodes and projects and 6 values of split ratio (2, 4, 6, 8, 10 and V).

Table 1. Parameters of tested systems

	Small systems	Large systems
Number of computing nodes	20	200
Number of projects	10	120
Cost of capacity module	120-400	120-400
Processing cost of one unit	50-150	50-150
Processing limit of one node	10-40	10-40
Number of destination nodes	1-4	1-8
Input and output data rates	5-15	5-10

The first part of experiments was devoted to tuning of the EA and TA algorithms. In Table 2, we report information about tuning parameters of the EA algorithm – for each parameter we present the tested values and the final selection of the parameter value according to performed experiments for both small and large systems.

Table 2. Tuning parameters of the EA algorithm

	Tested Values	Selected Value
<i>population_size</i>	50, 100, 200, 300, 400	100
<i>offspring_population_size</i>	125, 250, 500, 750, 1000	250
<i>mutation_probability_1</i>	0.0, 0.05, 0.01, 0.02, 0.04, 0.05	0.01
<i>mutation_probability_2</i>	0.0, 0.05, 0.01, 0.02, 0.04, 0.05	0.02
<i>individual_age</i>	3, 10, ∞	10
<i>increase_split_frequency</i>	10	10
<i>decrease_split_frequency</i>	50	50

In the case of the TA algorithm the tuning procedure was done separately for two sizes of systems and different values of the split ratio. Table 3 shows tested values regarding each parameter. In Table 4 we report the selected values of parameters.

Table 3. Tested values of tuning parameters of the TA algorithm

	Small systems	Large systems
<i>exploration_range</i>	10, 20, 30, 40, 50, 60, 70, 80, 100, 120, 150, 200, 300, 350, 400, 450, 500	10, 20, 40, 60, 80, 120, 160, 200, 250, 300, 400, 500, 600
<i>exploration_deep</i>	1 to 10	1 to 10
<i>tabu_list_type</i>	Equality, Similar	Equality, Similar
<i>tabu_list_length</i>	10 to 180	10 to 800
<i>search_method</i>	by project, by node	by project, by node
<i>aspiration_type</i>	better than best better than current	better than best better than current

Table 4. Selected values of tuning parameters of the TA algorithm

Split ratio	Small		Large	
	2, 4, 6	8, 10, 20	2, 4, 6	8, 10, 200
<i>exploration_range</i>	20	30	600	400
<i>exploration_deep</i>	3	8	1	1
<i>tabu_list_type</i>	Equality	Similar	Similar	Similar
<i>tabu_list_length</i>	60	50	20	20
<i>search_method</i>	by project	by node	by project	by node
<i>aspiration_type</i>	better than best	better than best	better than best	better than best

In Fig. 2, we report detailed results related to tuning of two parameters of the TA algorithm, i.e., exploration range and search method. The results were obtained for an exemplary large system with the split ratio set to 2. More results related to tuning of EA and TA algorithms can be found in [22] and [23], respectively.

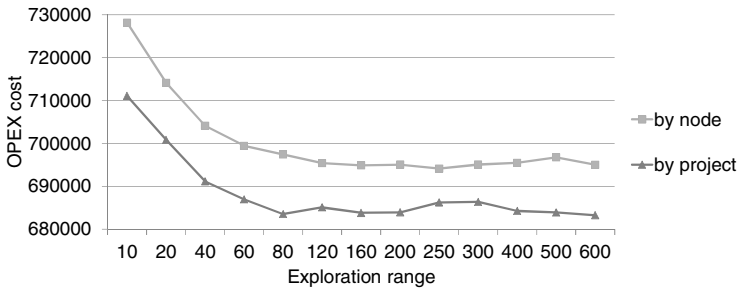


Fig. 2. Tuning of the TA algorithm – OPEX cost of a large system with split ratio 2 as a function of the exploration range and search method.

The next goal of experiments was to compare performance of developed heuristic algorithms described in Section 4 against CPLEX solver [21] that applies a branch-and-cut algorithm for ILP problems. In Table 5 we present comparison of heuristics against CPLEX for small and large systems. Note that EA and TA were run 3 times for each case and the average result was taken into account. The table includes average gap between particular heuristics and CPLEX as well as the lengths of 95% confidence intervals. Recall, that for each system size (small and large) 162 unique experiments were made.

Table 5. Heuristics vs. CPLEX – result comparison and lengths of 95% confidence intervals

	Small systems		Large systems	
	Average gap to the optimal results	Lengths of 95% confidence intervals	Average gap to the CPLEX results	Lengths of 95% confidence intervals
RA	22.93%	0.41%	21.97%	0.82%
GA	8.49%	0.31%	12.65%	0.60%
EA	2.99%	0.21%	10.28%	0.55%
TA	0.38%	0.05%	0.54%	0.88%

For small systems the CPLEX was run to find optimal results, i.e., the execution time of CPLEX was not limited, while the execution time of heuristics was maximally 20 minutes. We can easily notice that the best performance offers the Tabu Search algorithm – the average gap to optimal results is only 0.38%. For large systems the CPLEX could not provide optimal results in reasonable time, thus the execution time of CPLEX was limited to 20 minutes. Consequently, the results of CPLEX were not optimal. To enable fair comparison with heuristics, the running times of heuristic algorithms were also limited to 20 minutes. It should be noted that the RA needed about 10ms, the GA found the final results after 5 minutes, while EA and TA were stopped exactly after 20 minutes. Again, the TA method provides the best results among all heuristics. We can notice that performance of the EA in comparison with CPLEX decreased what suggests that the EA algorithm is not well scalable.

Since the TS significantly outperforms other heuristics, in the rest of this section we will focus on results of this algorithm. Table 6 reports detailed comparison between TS and CPLEX for large systems regarding the split ratio. We can notice that the TS method outperforms CPLEX for small values of split ratio (i.e., 2 and 4). These cases are the most challenging for optimization since only a very small part of the solution space is feasible in terms of the split ratio constraint (7). Other heuristics encountered large difficulties to find feasible results for small values of the split ratio.

Table 6. Tabu Search algorithm versus CPLEX – results obtained for large systems

Split ratio	2	4	6	8	10	200
Average gap between TA and CPLEX	-9.04%	-3.14%	1.86%	3.50%	4.08%	5.64%
Number of cases when TS>CPLEX	26	20	11	1	0	0

Due to stochastic nature of the TA method, we examined convergence of this algorithm. For selected small and large systems with all tested split ratio values we run the algorithm 20 times. The obtained results proved that the TS algorithm is very stable, the value of the standard deviation was always lower than 0.5% of the average value.

The last goal of simulations was to examine performance of the overlay computing systems according to the split ratio. In Fig. 3, we present the optimal cost of three exemplary small systems as a function of the split ratio value. We can easily notice that the OPEX cost decreases with the increase of the split ratio. However, the gap between two border cases (i.e., the difference in OPEX cost obtained for split ratio

values 2 and 20) is very small. Taking into account all tested small systems, this gap is on average 1.41%. The corresponding value for large systems is calculated for results of the TA algorithm is 1.58%. Thus, we can conclude that the OPEX cost is not significantly sensitive to the split ratio value. However, we must underline that the value of the split ratio had a large impact on the performance of evaluated algorithms in terms of the result feasibility and differences between particular methods.

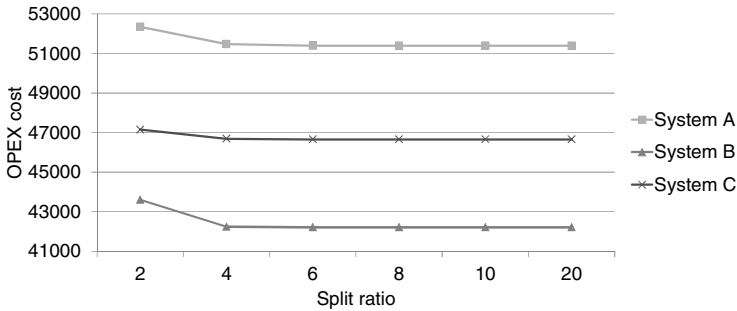


Fig. 3. OPEX cost of small systems as a function of the split ratio parameter

6 Concluding Remarks

In this paper we have described a scheduling and capacity design optimization problem in overlay computing systems. Whereas, most of previous papers on the topic of distributed systems optimization focus on dimensioning of optical networks, we have assumed that the considered system works in overlay mode. Due to increasing popularity of virtualization techniques following from cost-effective and flexible deployment comparing to traditional network model, overlay systems have been gaining much attention in recent years. The problem has been formulated in the form of ILP model. Next, four heuristic algorithms have been developed. In extensive numerical experiments the algorithms have been examined to find the best configuration of tuning parameters. Next, the heuristics have been evaluated in comparison to optimal solutions provided by the CPLEX solver for small problem instances. The best performance has been observed for the Tabu Search algorithm – the average gap to optimal results was only 0.38%. Moreover, for large systems examined with 20-minute execution time limit, the Tabu Search method outperformed all remaining methods including the CPLEX solver in the case of small values of the split ratio parameter.

In future work, we plan to consider further optimization problems encountered in overlay computing distributed systems considering additional constraints like survivability issues and more complex workflows.

Acknowledgement. The work was supported in part by the National Science Centre (NCN), Poland.

References

1. Wilkinson, B.: *Grid Computing: Techniques and Applications*. Chapman & Hall/CRC Computational Science (2009)
2. Nabrzyski, J., Schopf, J., Weglarz, J. (eds.): *Grid resource management: state of the art and future trends*. Kluwer Academic Publishers, Boston (2004)
3. Milojcic, D., et al.: *Peer to Peer computing*. HP Laboratories Palo Alto, Technical Report HPL-2002-57 (2002)
4. Travostino, F., Mambretti, J., Karmous Edwards, G.: *Grid Networks Enabling grids with advanced communication technology*. Wiley (2006)
5. Taylor, I.: *From P2P to Web services and grids: peers in a client/server world*. Springer (2005)
6. Buford, J., Yu, H., Lua, E.: *P2P Networking and Applications*. Morgan Kaufmann (2009)
7. Shen, X., Yu, H., Buford, J., Akon, M. (eds.): *Handbook of Peer-to-Peer Networking*. Springer (2009)
8. Tarkoma, S.: *Overlay Networks: Toward Information Networking*. Auerbach Publications (2010)
9. Stevens, T., et al.: *Multi-Cost Job Routing and Scheduling in Optical Grid Networks*. *Future Generation Computer Systems* 25(8), 912–925 (2009)
10. Thysebaert, P., et al.: *Scalable Dimensioning of Resilient Lambda Grids*. *Future Generation Computer Systems* 24(6), 549–560 (2008)
11. Develder, C., et al.: *Survivable Optical Grid Dimensioning: Anycast Routing with Server and Network Failure Protection*. In: *Proc. of IEEE ICC 2011*, pp. 1–5 (2011)
12. Kacprzak, T., Walkowiak, K., Woźniak, M.: *Optimization of Overlay Distributed Computing Systems for Multiple Classifier System – Heuristic Approach*. *Logic Journal of IGPL* (2011), doi:10.1093/jigpal/jzr020
13. Akbari, B., Rabiee, H., Ghanbari, M.: *An optimal discrete rate allocation for overlay video multicasting*. *Computer Communications* 31(3), 551–562 (2008)
14. Zhu, Y., Li, B.: *Overlay Networks with Linear Capacity Constraints*. *IEEE Transactions on Parallel and Distributed Systems* 19(2), 159–173 (2008)
15. Pioro, M., Medhi, D.: *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers (2004)
16. Michalewicz, Z.: *Evolutionary Algorithms + Data Structures = Evolution Programs*. Springer (1999)
17. Gendreau, M., Potvin, J.: *Handbook of metaheuristics*. Springer (2010)
18. Glover, F.: *Tabu Search - Part I*. *ORSA J. on Computing* 1(3), 190–206 (1989)
19. Karlsson, L., Yang, L., Lin, M.: *Heuristic techniques: scheduling partially ordered tasks in a multi-processor environment with tabu search and genetic algorithms*. In: *Proceedings Seventh International Conference on Parallel and Distributed Systems*, pp. 515–523 (2000)
20. Jaziri, W.: *Local Search Techniques: Focus on Tabu Search*. Inteh (2008)
21. ILOG CPLEX, 12.0 User's Manual, France (2007)
22. Miziolek, M.: *Optimization of distributed computing systems with the use of evolutionary algorithms*. M.Sc. Thesis, Wroclaw University of Technology (2011)
23. Kosowski, M.: *Optimization of distributed computing systems with the use of Tabu Search algorithm*. M.Sc. Thesis, Wroclaw University of Technology (2011)

GPU Acceleration of the `caffa3d.MB` Model

Pablo Igounet¹, Pablo Alfaro¹, Gabriel Usera², and Pablo Ezzatti¹

¹ Instituto de Computación,

Universidad de la República, 11.300–Montevideo, Uruguay

{`pigounet,palfaro,pezzatti`}@`fing.edu.uy`

² Instituto de Mecánica de los Fluidos e Ingeniería Ambiental,

Universidad de la República, 11.300–Montevideo, Uruguay

`gusera@fing.edu.uy`

Abstract. This work presents a study of porting Strongly Implicit Procedure (SIP) solver to GPU in order to improve its computational efficiency. The SIP heptadiagonal linear system solver was evaluated to be the most time consuming stage in finite volume flow solver `caffa3d.MB`. The experimental evaluation of the proposed implementation of the solver demonstrates that a significant runtime reduction can be attained (acceleration values up to 10×) when compared with a CPU version, and this improvement significantly reduces the total runtime of the model. This results evidence a promising prospect for a full GPU-based implementation of finite volume flow solvers like `caffa3d.MB`.

Keywords: `caffa` model, finite volume, SIP, HPC, GPU.

1 Introduction

The `caffa3d.MB` open source flow solver [15] allows the numerical solution of equations that govern the movement of viscous fluids in complex geometry. The model was implemented by Ferziger and Peric [6] using the original 2D `caffa` model as a baseline. Nowadays, `caffa3d.MB` is largely used by different laboratories and a typical instance, e.g. a simulation of 12 hours of stratified flow case with 5 million nodes, involving about 40.000 time steps, can be computed on about 300 hours, using a modern CPU with four cores. The source code of `caffa3d.MB` is publicly available at the website www.fing.edu.uy/imfia/caffa3d.MB/, and the model is composed by approximately 50.000 lines of code.

The high runtime involved on fluids simulation motivates the use of high performance computing (HPC) techniques. However, the traditional HPC techniques usually implies important economical efforts. A modern strategy consist in using secondaries processors for tackle general problems. Particularly in recent years, graphics co-processors (GPU) have experienced a spectacular evolution. The evolution was not only quantitative (i.e. computing power) but also has largely been qualitative (i.e. GPUs have improved in capacity and flexibility for programming). Since the release of CUDA (Compute Unified Device Architecture) [8] by NVIDIA in 2007, GPUs have become easily programmable multi-processors capable of executing parallel algorithms through the SPMD paradigm

(Single Program - Multiple Data). In fact, the GPU employs a SIMT (Single-Instruction, Multiple-Thread) a slightly different strategy, defined by NVIDIA. This situation has motivated many scientists from different fields to take advantage of the use of GPUs in order to tackle general problems [10] (databases, scientific computing, etc.). In particular, in the area of numerical linear algebra (NLA) several studies have been presented, showing significant improvements regarding performance by using GPUs [2,3,5,14]. A distinctive characteristic of the CUDA architecture is its memory hierarchy, which consists of several memory levels: registers, shared, local, global, constant and texture memory. Each of these levels have different sizes, bandwidth, throughput, etc. and the programmer has to explicitly indicate which memory levels are used. Additionally, CUDA offers various techniques to exchange data between CPU and GPU memories: common transfer, page locked transfer (using pinned memory) and direct access.

The importance of the numerical model, the high computational cost and the steady evolution of GPUs motivate this work, which presents the analysis of porting the `caffa3d.MB` model to GPU-based computation. An analysis of the computational performance of `caffa3d.MB` is performed in order to identify the most costly stages of the model. Later we present the GPU-based versions proposed for speeding-up the computation using GPUs. The preliminary results demonstrate that significant values of the acceleration factor (up to 24% for the whole model) are obtained when using a GPU card that costs less than a 10% of the price of a standard multicore server.

This paper is organized as follows. The study of the `caffa3d.MB` model can be found in Section 2, where the mathematical model, the iteration scheme and the evaluation of model runtime are detailed. Section 3 reviews related work about solving NLA problems on GPUs and the parallelization of SIP method and, later on, presents and explains the implementations developed. Finally, in Section 4, the conclusions and the future lines of work are discussed.

2 `caffa3d.MB` Model

The `caffa3d.MB` model implements the finite volume method, applied to the 3D numerical simulation of viscous and/or turbulent fluids with generic scalars transport. The domain geometry is represented by block-structured curvilinear grids, which allow to describe complex scenarios. The interface between blocks is fully implicit, in order to avoid downgrading both the performance and the numerical results of the method. The model allows incorporating rigid objects in the domain to study their interactions with the fluid, through immersed boundary condition method.

2.1 Mathematical Model

The mathematical model comprises the mass (1) and momentum (2) balance equations for an incompressible Newtonian fluid, with the Boussinesq approximation for buoyancy effects due to temperature induced small density variations:

$$\int_S (\mathbf{v} \cdot \hat{\mathbf{n}}_s) dS = 0 \tag{1}$$

$$\int_{\Omega} \rho \frac{\partial u}{\partial t} d\Omega + \int_S \rho u (\mathbf{v} \cdot \hat{\mathbf{n}}_s) dS = \int_{\Omega} \rho \beta (T - T_{ref}) \mathbf{g} \cdot \hat{\mathbf{e}}_1 d\Omega + \int_S -p \hat{\mathbf{n}}_s \cdot \hat{\mathbf{e}}_1 dS + \int_S (2\mu \mathbf{D} \cdot \hat{\mathbf{n}}_s) \cdot \hat{\mathbf{e}}_1 dS \tag{2}$$

These equations hold in any portion Ω of the domain, being S the boundary of Ω and $\hat{\mathbf{n}}_s$ the outward normal vector at the boundary S . The momentum balance equation (2) has been expressed for the first component u of the velocity vector $\mathbf{v} = (u, v, w)$, with similar expressions holding for the other components. The buoyancy term in equation (2) involves the density ρ , thermal expansion coefficient β and temperature T of the fluid, a reference temperature T_{ref} and gravity \mathbf{g} . The viscosity μ of the fluid and the symmetric deformation tensor \mathbf{D} are used in the viscous term.

The conservation law (3) for a generic passive scalar ϕ with diffusion coefficient Γ is also considered, which of course includes the heat equation as a particular case. Other scalar transport equations can be easily incorporated, for example to construct turbulence models or to address wet air processes which include evaporation and condensation.

$$\int_{\Omega} \rho \frac{\partial \phi}{\partial t} d\Omega + \int_S \rho \phi (\mathbf{v} \cdot \hat{\mathbf{n}}_s) dS = \int_S \Gamma (\nabla \phi \cdot \hat{\mathbf{n}}_s) dS \tag{3}$$

These equations were presented in their integral form, in compliance with the finite volume method and promoting a conservative formulation. The discretized equations will be obtained by applying equations (1, 2 and 3) to each volume element.

2.2 caffa3d.MB Iteration Scheme

In each time step, equations for each component of the velocity and equation for the pressure correction are solved alternately and successively. Equations for scalar transport, which will not be detailed here, are also solved. Thus, the solution to the non-linear, coupled, partial differential equation system (1, 2 and 3) is approximated by the solution of a succession of linear equation systems, as represented in Figure 1. The caffa3d.MB model currently employs, for the iterative solution of these linear systems, the SIP method [6]. The SIP algorithm has found widespread use in CFD, both as a solver itself or as a smoother for multigrid methods. For a complete description of caffa3d.MB please see [16].

2.3 Analysis of caffa3d.MB Model Runtime

Experimental Platform. The experimental evaluation platform consists of two different CPUs each connected to a different GPU. Table 1 presents details of the platforms used in the experimental evaluation. Note that E5530 processor use Hyper-Threading, that is 16 symbolic cores.

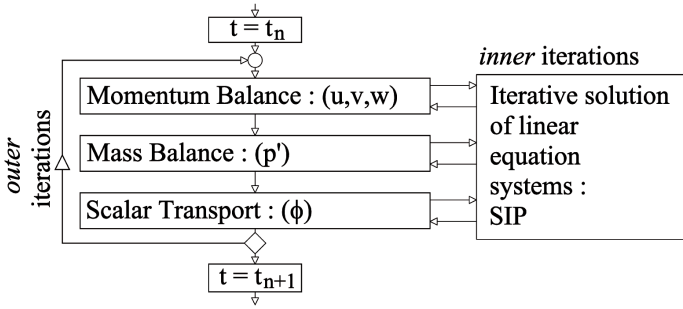


Fig. 1. Iteration scheme for one time step (adapted from [11])

Table 1. Platforms used for testing

Platform	CPU	Cores	GPU model	Multicores	Cores
I	Intel E5200 @ 2.5 GHz 2GB	2	9800 GTX+	16	128
II	Intel QuadCore E5530 2.4GHz 48GB	8	C1060	30	240

Test Cases. The application considered is the lid-driven flow inside a cubical cavity with $Re=1000$. This is a traditional test case for Navier-Stokes solvers, with good numerical solutions being available in the bibliography [17] to validate the obtained results. Non slip boundary conditions are applied to all walls, with the top one sliding at constant velocity V_o . Given the height h of the cavity and the lid velocity V_o , the fluid viscosity ν is set so that $Re = 1000$. The computations were run until steady state was achieved. Figure 2 presents a sketch of the domain.

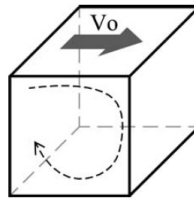


Fig. 2. Sketch of lid-driven cavity. $Re = V_o \cdot h/\nu = 1000$

We evaluated two test cases which differ on the grid dimension, one for a $64 \times 64 \times 64$ grid and another discretized in $128 \times 128 \times 128$ volumes, named `CAVS64` and `CAVS128` respectively.

Evaluation of `caffa3d.MB` Runtime. To perform the model evaluation only Platform II was used. We simulate 10 time steps of case `CAVS128` with the original model.

The experiment shows that three routines (*calcuw*, *gradfi*, *sipsolver*) are the most costly in runtime. Particularly, on a total runtime of 518 seconds the linear system solver using the SIP method consumed 35% (178 seconds), the *calcuw* routine that performs the discretization and resolution of the linearized equations for momentum components implied 25% (136 seconds) and the computation of the components of the gradient vectors with the *gradfi* routine took around 15% (79 seconds). Therefore, the computational cost of these three routines is roughly 75% of the total runtime of the model.

3 caff3d.MB on GPU

The runtime evaluation presented in the previous section shows that the time required to execute the SIP method impacts strongly in the computational efficiency of the `caff3d.MB` model. For this reason, we first studied and implemented a GPU-based version of the SIP method. This strategy of separating the most costly stage of a model was also employed in other similar efforts, see e.g. [9] a GPU-based version of WRF model.

3.1 SIP on GPU

SIP [13] is an iterative method for the resolution of band systems ($Ax = b$, where A is heptadiagonal or pentadiagonal matrix, related to discretization of differential equations). Initially, in the SIP method, an incomplete LU factorization ($\hat{L}\hat{U} = incLU(A)$) is computed, where \hat{L} and \hat{U} are good approximations of L and U matrices without fill-in.

After that, an iterative procedure for refining an initial solution is performed until the residual is small enough. The Algorithm SIP presents the SIP method.

Algorithm SIP:

- (1) Do incomplete LU decomposition: $\hat{L}\hat{U} \approx A$
- (2) Calculate residual: $r_n = b - Ax_n$
- (3) Calculate vector R_n (forward substitution):

$$R_n = \hat{L}^{-1}r_n$$
- (4) Calculate δx (backward substitution):

$$\hat{U}\delta x = R_n$$
- (5) Back to (2), until residual is small enough.

3.2 Related Works

Over the last years, different studies have demonstrated the benefits of using GPUs to accelerate the computation of general purpose problems and in particular of linear algebra operations. However, there are no proposals for a GPU-based SIP solver and no article was found discussing this subject in the review of related works.

Several works about SIP parallelization were presented using traditional HPC hardware. We highlight the work of [4], that studied the parallelization of the SIP method on shared memory multiprocessors considering different orderings to process the nodes for breaking the data dependencies. These orderings are described next, over a typical 3D-grid, such as shown in Figure 3, with a $n_i \times n_j \times n_k$ dimension.

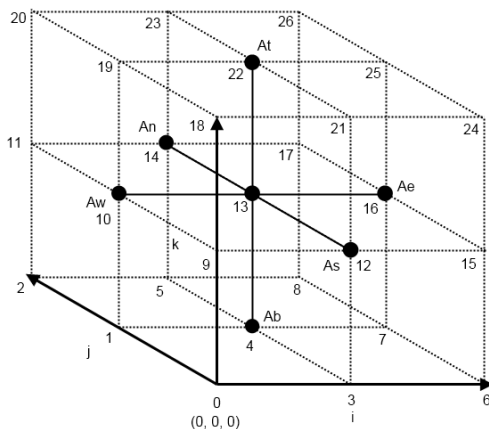


Fig. 3. An example 3D grid, showing the 7-node computation molecule

Sequential Ordering. This variant processes the coefficients sequentially, according to the number of each node. This strategy allows to exploit data locality on CPU.

Ordering by Hyperlines. Considering the data dependencies of the SIP method it can be established that:

- The residual calculation does not present restrictions.
- The forward substitution, only presents a dependency between r elements. Particularly, to compute node l , the values of nodes $l-1$, $l-n_j$ and $l-n_i \times n_j$ are needed.
- The backward substitution, only presents a dependency between r elements. Particularly, to compute node l , the values of nodes $l+1$, $l+n_i$ and $l+n_i \times n_j$ are needed.

So the lines formed by nodes where $j+k$ is constant can be computed in parallel mode for both, forward and backward substitution. Inside each line, the nodes must be computed in ascending order of i for forward substitution, and descending order for backward substitution.

Ordering by Hyperplanes. In addition to the parallelism attained by ordering the calculations by hyperlines, another level of parallelism can be reached if the

nodes are grouped by hyperplanes where $i + j + k$ is constant, this ordering does not present data locality, however it offers the largest parallelism level.

3.3 Our Proposal

We implement three variants of the hyperplanes SIP method, a CPU implementation and two variants for GPU (a simple initial version and another which exploits coalesced access).

Each routine receives as input the seven diagonals ($A_p, A_s, A_w, A_b, A_n, A_e$ y A_t , following the geographic nomenclature used by Ferziger and Peric [6]), the α parameter, the independent term, and the maximum number of refining iterations; and solves the linear system, returning the solution vector, the number of iterations performed and the final residual. All routines use norm-1 to calculate the residual.

Computing the hyperplane ordering is a relatively complex stage, and so the ordering in which the nodes are calculated, as well as other required values (i.e. the amount of hyperplanes employed), are computed on CPU prior to resolving the system.

Differences between the versions are discussed next.

SIP on CPU by Hyperplanes (SIP_{CPU}-HP). This variant processes the nodes on CPU, grouping them by hyperplanes and taking advantage of multicore parallel processing. SIP_{CPU}-HP is implemented in C using OpenMP.

SIP on GPU by Hyperplanes (SIP_{GPU}-HP). This variant processes the nodes on GPU grouping them by hyperplanes. The principal stages of this variant are (see Figure 4 for a diagram of the algorithm):

1. First the input values are copied from the CPU to the GPU.
2. Later, the solver is executed on the GPU.
3. Finally, the outputs are copied from the GPU to the CPU.

To perform the solver in GPU, each thread processes one node of the grid (corresponding to a matrix equation). As we explained before, this calculation is complex to implement independently on each node, so we first stored the processing order in a vector and then copy it to the GPU memory. On each iteration we have the hyperplanes offset as input and threads access to the vector keeps the processing order for determined which equation need resolve as shown in Figure 5

The number of threads and blocks employed on kernels are computed following the next recipe:

- if $\#nodes \leq \#threadsPerWarp$
1 block with $\#nodes$ threads
- if $\#nodes/\#threadsPerWarp < \#multiprocessors$
 $\#nodes/\#threadsPerWarp$ blocks with $\#threadsPerWarp$ threads

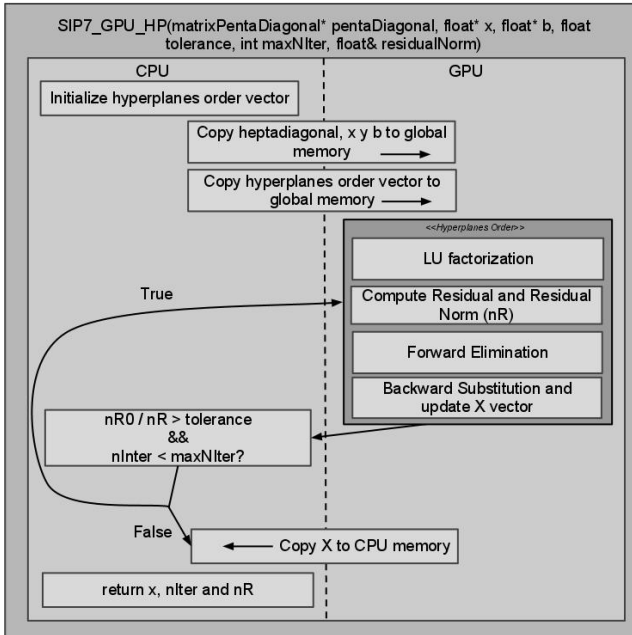


Fig. 4. Diagram of `SIPGPU-HP` version

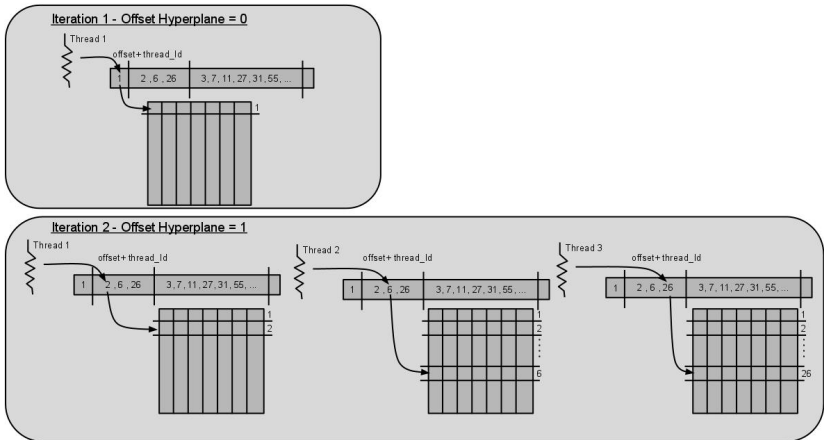


Fig. 5. Structure for computer processing

- if $\#nodes / \#threadsPerWarp > \#multiprocessors$
 - if $\#nodes < \#multiprocessors \times \#threadPerBlockLimit$
 $\#multiprocessors$ block with $\#nodes / \#multiprocessors$ threads
 - else
 $\#nodes / \#threadsPerBlockLimit$ blocks with $\#threadPerBlockLimit$ threads

SIP on GPU by Hyperplanes Using Coalesced Access (SIP_{GPU}-HPC). This variant extends the previous version in order to exploit coalesced access to memory. The forward and backward steps can be rearranged to achieve coalesced access, however, to achieve this goal, it is necessary to rearrange the A matrix, and as that step would increase the computational cost, therefore we only applied coalesced access to the calculation of the residual and norm computation. In Figure 6 is specified which order is used in each stage.

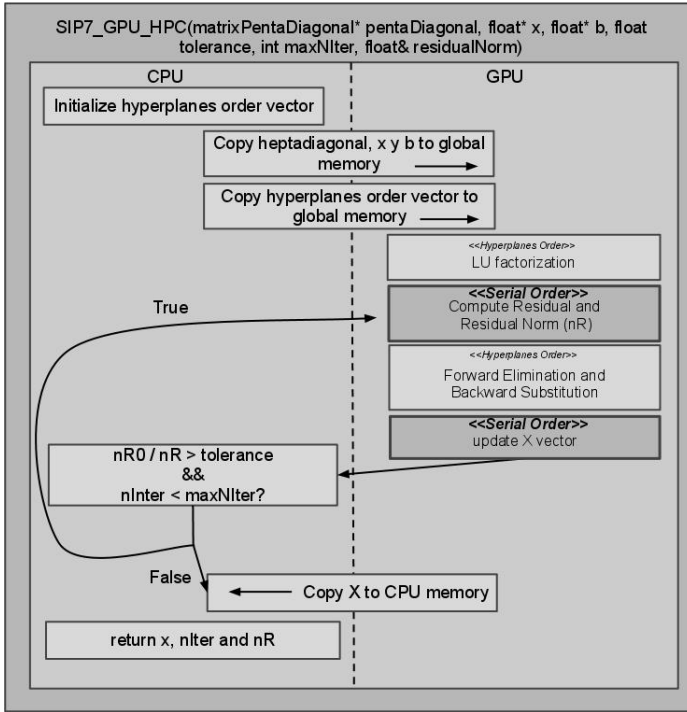


Fig. 6. Diagram of SIP_{GPU}-HPC version

3.4 Experiments

All experiments were performed using single precision arithmetic and transfer times are always included in the reported total runtime of the algorithm.

Tables 2 and 3 present the execution times (in seconds) for the CPU and GPU implementations of the SIP solver for heptadiagonal matrices of CAVS₆₄ and CAVS₁₂₈ cases on Platform I and II, respectively. Additionally, the tables include the acceleration values computed over the CPU method.

The analysis of the results demonstrate that the SIP_{CPU}-HP routine takes almost 3.86 and 1.57 times longer than the SIP_{GPU}-HPC routine on Platform I and Platform II respectively. In addition to this, the obtained acceleration scales on problem dimension, allowing 10.32× and 1.64× on each platform for the

Table 2. Runtime of SIP implementations on Platform I

Grid	SIP _{CPU} -HP	SIP _{GPU} -HP	SIP _{GPU} -HPC	Best Acceleration
$64 \times 64 \times 64$	0.37	0.097	0.096	3.86
$128 \times 128 \times 128$	6.251	0.622	0.606	10.32

Table 3. Runtime of SIP implementations on Platform II

Grid	SIP _{CPU} -HP	SIP _{GPU} -HP	SIP _{GPU} -HPC	Best Acceleration
$64 \times 64 \times 64$	0.099	0.081	0.063	1.57
$128 \times 128 \times 128$	0.592	0.535	0.362	1.64

largest case. It should be noticed that Platform II has 8 CPU cores (and Hyper-Threading) whereas Platform I has only 2, hence the difference in CPU runtimes and in speed up values.

Another important aspect, is that the use of coalesced access in Platform I produces a modest improvement on the runtime. This can be motivated because this platform has compute capabilities 1.1 and the coalesced access is more limited on such devices.

The execution time of different logical stages of SIP_{GPU}-HPC version of SIP was computed and analyzed in order to identify the critical section of the code. Table 4 presents the runtime (in seconds) of SIP_{GPU}-HPC version discriminated by transfer time, factorization time and resolution time. The obtained results show that the runtime for data transfer, factorization and each step of the iterative resolution procedure are similar (the resolution time in the table is for ten steps).

Table 4. Runtime of SIP implementations on Platform II

Grid	Transfer	Factorization	Resolution	Total
$128 \times 128 \times 128$	0.025	0.033	0.298	0.362

We can conclude that GPU versions have better performance than the CPU implementation, allowing to tackle larger problems with lower economic cost.

3.5 Integration of the GPU-SIP on the `caffa3d.MB` Model

We included the new GPU-based solver on `caffa3d.MB` model. In order to achieve this objective, we implemented a wrapper to call the new routine from the `caffa3d.MB` model. Note that the numerical model is implemented on Fortran95 and the solver routine on C and CUDA.

Additionally to the use of the wrapper, other minor modifications were included to the GPU SIP solver:

- The vector with processing order is copied only one time, at the beginning of model execution.
- Since in each step the `caffa3d.MB` model solve 3 different systems with the same matrix A , the matrix A is copied only one time each of the three systems solved.
- For the same reason, the LU factorization is computed only one time for each of the three systems solved.

Experiments. This section presents the results obtained in the evaluation of the GPU-based version of the `caffa3d.MB` model. The analysis was performed on Platform II solving the `CAVS64` and `CAVS128` test cases. The numerical results obtained with both model versions (CPU and GPU-based) not showed significant differences.

Table 5. `caffa3d.MB` model runtimes in seconds on Platform II

cases	Original <code>caffa3d.MB</code>	GPU-based <code>caffa3d.MB</code>	% of acceleration
<code>CAVS₆₄</code>	66.64	62.27	7 %
<code>CAVS₁₂₈</code>	518.33	418.92	24 %

The results in Table 5 demonstrate that good acceleration factor values can be obtained when using the parallel GPU implementation of the `caffa3d.MB` model. The acceleration factor values increase, achieving up to 24 % for the largest scenario.

It should be noted that the cost of data transfers between CPU and GPU is considered in the runtimes of our proposal. However, in a numerical model executed completely in GPU these transfers would not be necessary, allowing for better performances.

3.6 Other Model Stages on GPU

Such as the SIP solver was migrated to GPU, there is also the possibility of porting other stages of the model, since other calculations performed by the model can benefit from a SIMT architecture such as the one available on graphics processors.

In this way runtimes can be improved not only by accelerating other stages of the model, but also by eliminating the cost of constantly copying data between CPU memory and GPU memory. In this manner input data would be sent to the GPU only at the beginning of the whole process, and results would be copied back to CPU only at the end of the final calculation step.

To evaluate the parallelization potential of other stages of the model the *gradfi* routine was implemented on GPU. The *gradfi* routine computed the components of the gradient vectors for all discretized volumes. It was parallelized using CUDA, having each volume computed by one CUDA thread. Non formalized experiments show that an acceleration of around $5\times$ on computing time (not including transfer time) can be attained with the new GPU version of *gradfi* routine.

This preliminary result allows us to predict that porting the overall model to GPU-based computation can reach dramatically acceleration values.

4 Conclusions and Future Work

This work has presented an initial study on applying GPU computing in order to speed up the execution of the SIP method. A version of the SIP method was implemented on GPU using CUDA and was evaluated on different platforms with several matrices of increasing sizes. The experimental analysis showed that the GPU implementation reduced significantly the runtime of the solver over a parallel CPU implementation, e.g. `SIPGPU-HPC` version obtained an acceleration value up to $10\times$ on Platform I and 1.64 on Platform II. Additionally, the impact of this improvement on the total runtime of the complete model is important, attaining an acceleration of around 24%. Summing up, results show that the `caffa3d.MB` model can be accelerated with this kind of low economical cost platforms. Furthermore, the implemented version of SIP method on GPU might be easily adapted as the solver for other finite volume methods similar to `caffa3d.MB` model, and will be made shortly available in open source format together with next `caffa3d.MB` release.

Future research lines resulting from this experience will include:

- Tackling large problems.
- Using top line GPUs to further reduce computational times and increase the dimension of the affordable problems.
- Advancing on the study of a full GPU-based implementation of the model.
- Studying the SIP implementation with an interoperable tool, such as OpenCL.

Acknowledgements. The authors would like to thank Martín Pedemonte for his help. The equipment employed was financed by CSIC (Comisión Sectorial de Investigación Científica – Uruguay). Pablo Ezzatti and Pablo Igounet acknowledges support from Programa de Desarrollo de las Ciencias Básicas, and Agencia Nacional de Investigación e Innovación, Uruguay.

References

1. Albensoeder, S., Kuhlmann, H.C.: Accurate three-dimensional lid-driven cavity flow. *Journal of Computational Physics* 206, 536–558 (2005)
2. Barrachina, S., Castillo, M., Igual, F., Mayo, R., Quintana-Ortí, E., Quintana-Ort, G.: Exploiting the capabilities of modern GPUs for dense matrix computations. *Concurrency and Computation: Practice and Experience* 21, 2457–2477 (2009)

3. Benner, P., Ezzatti, P., Quintana-Ortí, E.S., Remón, A.: Using Hybrid CPU-GPU Platforms to Accelerate the Computation of the Matrix Sign Function. In: Lin, H.-X., Alexander, M., Forsell, M., Knüpfer, A., Prodan, R., Sousa, L., Streit, A. (eds.) Euro-Par 2009. LNCS, vol. 6043, pp. 132–139. Springer, Heidelberg (2010)
4. Deserno, F., Hager, G., Brechtefeld, F., Wellein, G.: Basic optimization strategies for cfd-codes. Technical report, Regionales Rechenzentrum Erlangen (2002)
5. Ezzatti, P., Quintana-Ortí, E., Remón, A.: Using graphics processors to accelerate the computation of the matrix inverse. *The Journal of Supercomputing* 58(3), 429–437 (2011)
6. Ferziger, J., Peric, M.: *Computational methods for fluid dynamics*. Springer, Berlin (2002)
7. Iwatsu, R., Hyun, J.M., Kuwahara, K.: Analyses of three dimensional flow calculations in a driven cavity. *Fluid Dynamics Research* 6(2), 91–102 (1990)
8. Kirk, D., Hwu, W.: *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann (2010)
9. Michalakes, J., Vachharajani, M.: GPU Acceleration of Numerical Weather Prediction. *Parallel Processing Letters* 18(4), 531–548 (2008)
10. Owens, J.D., et al.: A Survey of General-Purpose Computation on Graphics Hardware. *Computer Graphics Forum* 26(1), 80–113 (2007)
11. Peric, M.: *Numerical methods for computing turbulent flows* Course notes (2001)
12. Rhie, C.M., Chow, W.L.: A numerical study of the turbulent flow past an isolated airfoil with trailing edge separation. *AIAA Journal* 21, 1525–1532 (1983)
13. Stone, H.: Iterative solution of implicit approximations of multidimensional partial differential equations. *SIAM Journal of Numerical Analysis* 1(5), 530–558 (1968)
14. Tomov, S., Dongarra, J., Baboulin, M.: Towards dense linear algebra for hybrid GPU accelerated manycore systems. MIMS EPrint, 7, Manchester Institute for Mathematical Sciences, University of Manchester, Manchester, UK (2009)
15. Usera, G.: *caffa3d.MB User manual*, <http://www.fing.edu.uy/imfia/caffa3d.MB/caffa3d.MB.doc.pdf> (accessed November 20, 2011)
16. Usera, G., Vernet, A., Ferré, J.A.: A Parallel Block-Structured Finite Volume: Method for Flows in Complex Geometry with Sliding Interfaces. *Flow, Turbulence and Combustion* 81, 471–495 (2008)

Security-Effective Fast Authentication Mechanism for Network Mobility in Proxy Mobile IPv6 Networks

Illkyun Im, Young-Hwa Cho, Jae-Young Choi, and Jongpil Jeong

College of Information & Communication Engineering, Sungkyunkwan University
Seoul, Korea 110-745, +82-31-299-4260

illkyun.im@samsung.com, choyh2285@naver.com,
{jychoi, jpjeong}@ece.skku.ac.kr

Abstract. This paper reinforced security under the network evaluation of wire/wireless integration of NEMO (NEwork MObility) supporting mobility and network-based PMIPv6 (Proxy Mobile IPv6). It also proposes SK-L²AM (Symmetric Key-Based Local-Lighted Authentication Mechanism) based on simple key which reduces code calculation and authentication delay costs. Moreover, fast handoff technique was also adopted to reduce handoff delay time in PMIPv6 and X-FPMIPv6 (eX-tension of Fast Handoff for PMIPv6) was used to support global mobility. In addition, AX-FPMIPv6 (Authentication eXtension of Fast Handoff for PMIPv6) is proposed which integrated SK-L²AM and X-FPMIPv6 by applying Piggybacks method to reduce the overhead of authentication and signaling. The AX-FPMIPv6 technique suggested in this paper shows that this technique is better than the existing schemes in authentication and handoff delay according to the performance analysis.

Keywords: AAA, NEMO, MIPv6, HMIPv6, PMIPv6, Symmetric Cryptosystem, Hash Function.

1 Introduction

IETF (Internet Engineering Task Force) which is developing the Internet standard proposes the network class solution called as NEMO (NEtwork MObility) [1]. This enables the network to move among other external networks to maintain a continuous network connection through the expansion of MIPv6 (Mobile IPv6) [2]. But, NEMO inherited the disadvantage of handoff latency from MIPv6. Further, the method of processing AAA (Authorization, Authentication, and Accounting) in mobile technology network has not been defined. This implies that the upper network or lower network does not satisfy the stability and security. Therefore, the research for enhancing effectiveness accompanied by mobility is required, as secure authentication and fast handoff are accomplished.

This paper proposes SK-L²AM (Symmetric Key-Based Local-Lighted Authentication Mechanism) which is less burden some for the mobile device. Light

and local authentication was completed based on wire/wireless integrated network environment where NEMO supports mobility and network-based PMIPv6 (Proxy Mobile IPv6). Calculation cost and authentication delay factor is also considered in the proposed SK-L²AM. This has the following characteristics as. (1) Calculation cost is low. As SK-L²AM is the light weight security mechanism which uses only symmetric cryptosystem and hash function [3] to solve the problem of high calculation cost of PKI (Public Key Infrastructure). (2) As SK-L²AM provides local authentication, authentication delay gets reduced and load of HAAA (Home AAA) server declines as MR (Mobile Router) and LAAA (Local AAA) do not share the session key in advance. (3) SK-L²AM satisfies the security requirement by creating a session key, such as replay attack resistance, stolen-verifier attack resistance and mutual authentication to prevent server spoofing attack.

Besides, to reduce handoff delay under movement within a domain and among domains, it is extended so that the neighboring link layer address and the neighboring router information are easily processed. This is done by applying ND (Neighbor Discovery) protocol [4] in MAG (Mobile Access Gateway) at prior handoff and by supporting the advantages of FPMIPv6 (Fast Handoffs for PMIPv6). Global mobility is not supported in the network-based PMIPv6. In addition, AX-FPMIPv6 (Authentication eXtension of Fast Handoff for PMIPv6) which integrated SK-L²AM and X-FPMIPv6 by applying piggybacks method to reduce of telecommunication overhead is suggested. The AX-FPMIPv6 technique proposed in this paper shows that it is better than the existing scheme in authentication and handoff delay in the performance analysis.

The rest of this paper is composed as following. In chapter 2, security, handoff and integrated network architecture where PMIPv6 and NEMO are described. In chapter 3, the movement procedure of the proposed AX-FPMIPv6 authentication mechanism is explained in detail. Security analysis is evaluated in chapter 4 and AX-FPMIPv6 technique is evaluated based on performance evaluation measures in chapter 5. The research results are summarized in chapter 6.

2 Related Work

2.1 Secure Authentication

Many researches on handoff at movement in MIP (Mobile IP) environment as well as related to authentication using AAA are underway. This is to reduce the threat on the mobile network environment through secure authentication. Researches on AAA authentication are mostly concentrated on host mobility environment [5,6]. IETF proposes AAA model [7,8,9] and Diameter protocol [10] to solve problems when network is requested of roaming in external network from mobile node. As in AAA model (Fig. 1), there are 4 SA (Security Association) relationships in MIPv6. These SA means two network entities share several secret information with each other. When MR moves in domains, MR should provide several authentication information before it accesses resources from the domain.

But as in Fig. 1, direct security connection between MR and LAAA is insufficient and any information can be shared in advance between MR and LAAA which requires roaming and traditional authentication mechanism as one of technological problems. LAAA should resend information to HAAA (Home AAA) server of MR and wait for the response when there is no information for LAAA to confirm the authentication information. So, authentication becomes ineffective. Further, if MR often roams in different domains then, the authentication loads of MR get increased and it becomes more serious when the distance between the external network and the home network grows apart. NEMO does not specify how to process AAA in a mobile network and only a few of the researches have considered AAA authentication in NEMO environment. Fathi et al. [11] uses AAA model to deal with the security problems in NEMO. Here, the author proposes LR-AKE (Leakage-Resilient Authenticated Key Exchange) [12] system. It is based on the concept of PKI (Public Key Infrastructure). PKI can be used to prevent attacks in general but in this method calculation cost is excessive. This is shown in Wang et al. [13] that LR-AKE system is vulnerable to client and server impersonation attack. Chuang et al. [14] proposes local authentication concept to reduce the authentication delay. But, MR still requests for authentication from the AAA server again, when it moves initially to a new domain. Also, as it only supports local mobility, it should be registered as it enters a new domain for global mobility. Fig. 1 is a security association diagram of AAA model under PMIPv6.

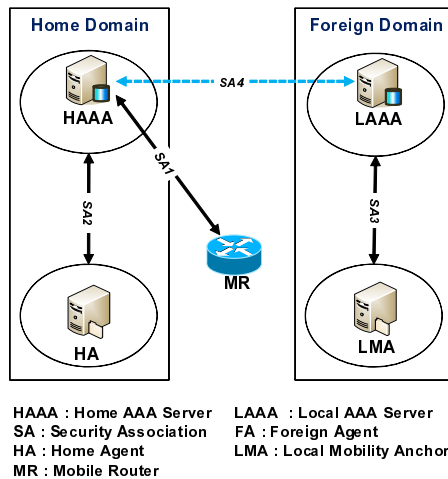


Fig. 1. Security association diagram of AAA model and PMIPv6

2.2 Network Architecture

Wireless network has at least one or more MR which is responsible for maintaining MNNs sessions. Besides, a domain should have several MAGs and one or more LAAA servers. When a wireless network enters into a new domain, MR

should execute the first authentication procedure before it accesses to the new network. Network phase often gets changed when the MNNs accesses, cuts and executes handoff. It is a very important task of a wireless communication to maintain effective secure group communication. The network architecture displayed in Fig. 1 can support all kinds of group key management systems of mobile communication network [15,16,17]. If a roaming agreement is made then, it should be noted that HAAA and LAAA servers share several secret information in advance to facilitate authentication procedure. This is based on AAA security model (SA4 in Fig. 1). Further, LAAA and MAGs also share common secret information such as GK (Group Key) (SA3 in Fig. 1). That is why LAAA and MAGs have security association.

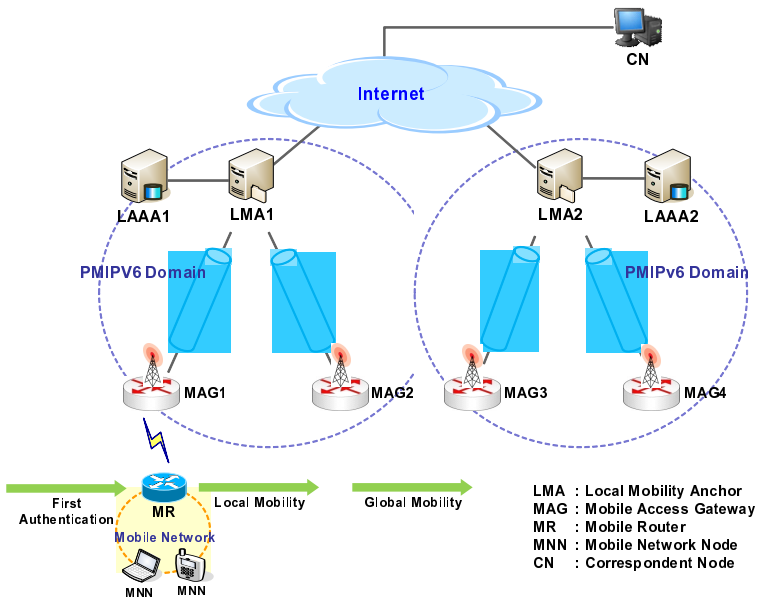


Fig. 2. Network architecture which combines AAA model under NEMO and PMIPv6

2.3 Fast Handoff in PMIPv6

As NEMO was extended from MIPv6, it inherited the disadvantage of long handoff delay. To solve this, many approaches [18,19,20] attempted to improve the long handoff delay of MIPv6. These systems also still possess certain disadvantages. For example, while Malki [18] proposes LLH (Low Latency Handoff) based on the prior registration method, MNN may cause transmission failure due to HER (HA Error Registration) problem under Ping-Pong movement. RFC 4068 [19] and RFC 4260 [20] propose FMIPv6 system in order to improve handoff performances of MIPv6. FMIPv6 depends on AR (Access Router) for handoff but there is no guarantee that MN is connected to AR every time. While supporting fast handoff among domains if access between the expected MN and AR fails then, access should be tried again. This can lead to long handoff latency. That

is why FMIPv6 uses simple 2 layer trigger. Also, MN requires protocol stack to process signaling for MN in MIPv6. This causes technological difficulties for supporting MIPv6 to limited MN, excessive resource, battery problem, etc. and they are turning out to be a hindrance for commercialization of terminals which supports MIPv6.

Therefore, NetLMN (Network-based Localized Mobility Management) WG of IETF standardized PMIPv6 as the network-based mobility management protocol. This guarantees service continuity under movement without MN modification by managing network for the MN's IP mobility to solve MIPv6 problems [21]. This implies that the MN may not have any capability for providing mobility service in PMIPv6. But for all the packets that are delivered to MN in PMIPv6 through LMA (Local Mobility Anchor), the packet bottleneck problem arises in LMA. There is a problem that continuity is not guaranteed in movement among PMIPv6 domains as local movement was considered from the initial stage.

So, while IETF proposed various methods such as Giaretta [22] method which supports mobility among the domains through hierarchical interface of MIPv6-PMIPv6 in order to support global mobility in NetLMN WG. Na's method [23] supports global mobility by defining additional signaling message among PMIPv6 domains, etc. The method of supporting handoff among domains through MIPv6-PMIPv6 interface has a problem that is the MN should have MIPv6 protocol stack and Na method causes handoff latency due to additional signaling message.

3 Security-Effective Fast Authentication Mechanism

3.1 Symmetric Key-Based Local-Lighted Authentication Mechanism

In this section, SK-L²AM (Symmetric Key-based Local-Lighted Authentication Mechanism) based on AAA model indicated in Fig. 1 is explained. There are 3 kinds of procedures for operating SK-L²AM such as home registration, the first registration in a domain and then, re-authentication registration. MR should be registered in the HAAA server before accessing into an external network. When MR first comes into an external network, SK-L²AM performs the first authentication procedure. When MR moves within the same domain, SK-L²AM executes fast re-authentication. And then, X-FPMIPv6 which is extended and improved with fast handoff technique in order to support handoff latency reduction within domains and global mobility which is not supported by PMIPv6 by reducing signaling overload is also proposed. Lastly, SK-L²AM is integrated into X-FPMIPv6 so that signaling overload is not increased. The notation in the proposed scheme are explained in Table 1.

MR should be executed by the home registration procedure before connecting MR into network. This registration can be made just by the security channel or by the manual work done by people. Let's assume there is security channel between MR and AAA based on Diameter protocol in AAA model and Diameter protocol which is explained above. The reason for security association between MR and HAAA can be seen from Fig. 1. If there is no security association

Table 1. Notation

Symbol	Description
χ	Secret value which is shared between HAAA and LAAA
GK	Group key of domain
MAC_i	Unique MAC address which I, a mobile communication device
R_i	I, an arbitrary value
$E_K(M)$	Coded message by using symmetric code (Encryption) K key
$D_K(M)$	Plain sentence is decoded in symmetric key K
H()	Hash function of work direction open
	Character string combination
$X_{service}$	All accessing rights in HAAA server
$Y_{service}$	$Y_{service} \subseteq X_{service}$: A set of accessing rights of MR
$Z_{service}$	$Z_{service} \subseteq Y_{service}$: Accessing right which LAAA allowed
SK	Session Key

between MR and HAAA then, the system can execute Diffie-Hellman system to establish security channel.

When MR enters into a new network, MR executes the first authentication procedure. As MR often moves to other domains in a wireless communication network, it should perform the re-authentication operation. Generally, the authentication information of MR should be confirmed from the HAAA server. If a domain is far from the home domain then, it will take a long time for authentication. Therefore, an effective authentication mechanism is required. The system proposed here provides a local authentication mechanism (Namely, it can authenticate in local without including a remote server) and it can facilitate a mutual authentication between MR and LAAA server.

When MR moves within the same domain from the existing MAG to another MAG, it should receive confirmation again. As in Fig. 3, MAGs executes fast re-authentication procedure. In the first authentication and fast re-authentication procedures, SK-L²AM provides local authentication. In the first authentication procedure, MR uses G key to generate in the home registration procedure to achieve local authentication. LAAA server substitutes HAAA server to execute the authentication procedure. That is why G key can be calculated from the secret value χ which is shared by LAAA server and HAAA server. Similarly, in the fast re-authentication procedure, MR uses K key which is achieved from LAAA server in the first authentication procedure. As K key is generated between external network LAAA server and MAGs, MAG executes authentication procedure instead of LAAA server to reduce authentication latency.

In key management, cost of symmetric cryptosystem proposed in this paper is very low. Although it is based on symmetric cryptosystem, all entities need to save a few of the parameters (i.e., in case of HAAA and LAAA, secret value χ and

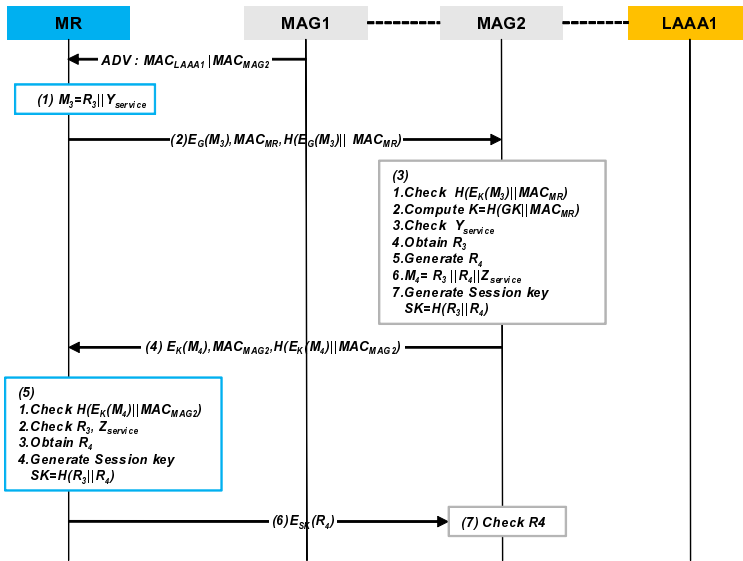


Fig. 3. Re-authentication procedure in the same domain

access right of MR). This is the reason for the LAAA to calculate symmetric G key to generated by $H(\chi \parallel MAC_{MR})$ on time rather than finding the applicable G symmetric key through key management in the first authentication procedure.

3.2 eXtension & Fast Proxy Mobile IPv6 (X-FPMIPv6)

This paper proposes X-FPMIPv6 which is extended and improved from PMIPv6. This is done to enable fast handoff process by achieving neighboring L2 layer and neighboring router information (Namely, including LMAs || MAGs and network prefix information). ND protocol is also applied in MAG to support global mobility. Moreover, fast handoff is also applied at handoff among domains.

Fig. 4 shows the procedure of X-FPMIPv6 proposed when MR moves within the same domain. Especially, X-FPMIPv6 added two L2 trigger (prior L2 trigger, p-LT(pre-Link Trigger) and start L2 trigger, s-LT(start-Link Trigger) for the execution of pre-handoff procedure in advance, and also to reduce the handoff latency and provide secure handoff. p-LT operation starts when the signal strength received from MAG is lower than the threshold designated in advance in MR. This is to extend the concept of "DeuceScan" [24] and to avoid the performance decrease due to Ping-Pong effect of X-FPMIPv6 when MR gets apart from the previous MAG. The start of s-LT means that MR started the procedure of handoff in advance. X-FPMIPv6 uses two triggers for interaction between L2 and L3. The two triggers of p-LT and s-LT provide more accurate information to reduce the possibility of handoff failure.

To summarize this, X-FPMIPv6 supports network layer handoff procedure and uses many triggers L2 to avoid HER problems. Further, as it completed its

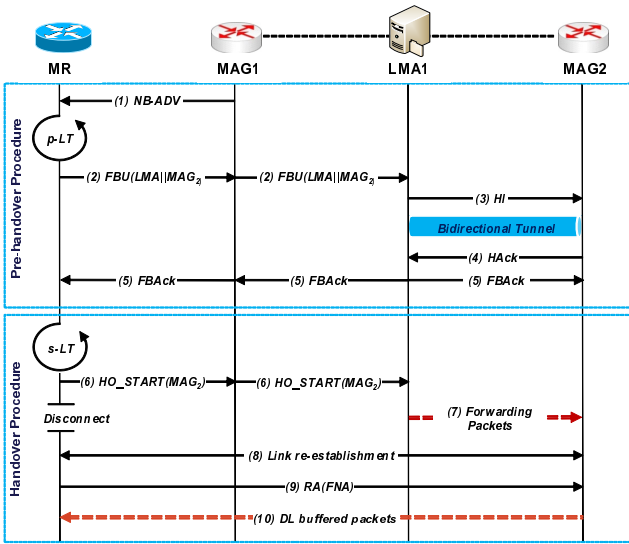


Fig. 4. Fast handoff procedure within X-FPMIPv6 domain

movement search and pre-handoff procedure, the proposed X-FPMIPv6 reduces handoff latency. After pre-handoff procedure, MR has many PCoAs simultaneously. So, even if MR makes a wrong handoff decision, it can be connected to new MAG immediately if handoff arises.

PMIPv6 does not support global mobility. But this paper improved the advantages of fast handoff method and realized handoff procedure among the domains by the application of fast PMIPv6. 1~5 stages are to execute pre-handoff procedures and 6~10 stages are real handoff procedure.

To summarize fast handoff in global mobility, the pre-handoff is prepared by detecting the neighboring MAGs and in case of moving MAGs in the same domains; the fast handoff procedure (Fig. 4) is executed. But if not MAGs in the same domain, X-FPMIPv6 reduces handoff latency even under global mobility as it completes pre-handoff by delivering HI simultaneously to candidate MAGs of other domains and LMAs to which they belong.

3.3 Integrated Operation of SK-L²AM and X-FPMIPv6 (AX-FPMIPv6)

As AX-FPMIPv6 aims at the reduction of handoff latency in domain changing regardless of local or global mobility by MR, it should integrate with proper authentication systems. If the authentication procedure is executed in home domain then, the design of AX-FPMIPv6 should be discarded. In fact, SK-L²AM operates along with X-FPMIPv6 while supporting local authentication.

In this section, AX-FPMIPv6 reinforces security in local and global mobility and it reduces handoff latency with light/fast SK-L²AM and signaling overload as it describes, how fast the authentication procedure is performed under mobile

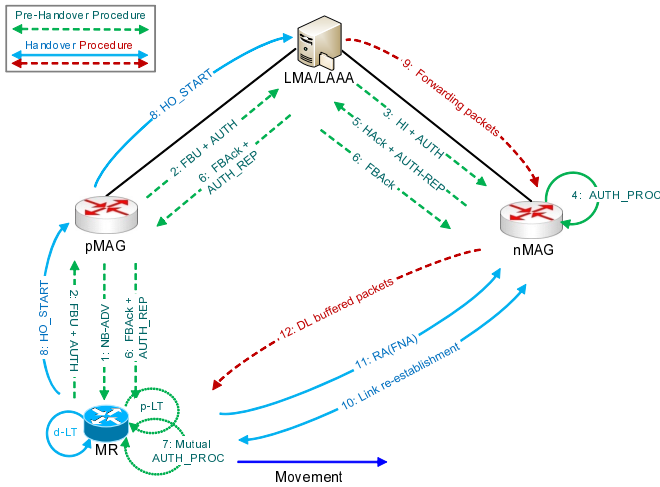


Fig. 5. AX-FPMIPv6 sequence diagram within the same domain

environment. This supports fast re-authentication procedure where MR supports global mobility in the same domain while X-FPMIPv6 operates in integration. Fig. 5 comprises of pre-handoff and real handoff procedure which represents the integration of handoff and authentication operation in the same domain.

When MR sends FBU message to pMAG(previous MAG) of MR, authentication message (AUTH) is transmitted by Piggybacks method and MAC_{MR} and $M(R_3 \parallel Y_{service} \parallel H(R_3 \parallel MAC_{MR}))$ which is expressed in Fig. 3 are included. And if MR is proved to be effective at this time then, nMAG(new MAG) transmits authentication reply message (AUTH-REP) by Piggybacks method as a reply of HAck/FBack for the MR. Moreover, MAC_{MAG} and $E_K(R_3 \parallel R_4 \parallel Z_{service} \parallel H(R_4 \parallel MAC_{MAG_2}))$ are included in the message, reduction of signaling overload and secure/seamless communication is possible as SK-L²AM and X-FPMIPv6 are operating in integration. During movement among domains, AX-FPMIPv6 where signaling and authentication message (AUTH) is integrated is as shown in Fig. 5 and it operates using Piggybacks method. But if it is not MAG which is currently accessed but nMAG of a new domain then, HI and AUTH are delivered to the nMAG and nLAAA(new LAAA) respectively. As for the fast reply for them, pLMA(previous LMA) and pMAG transmit FBack and (AUTH-REP) to pLMA. This reduces handoff latency and signaling overload and provides safe and secure/seamless communication by the operation of SK-L²AM and X-FPMIPv6 with the integration of authentication and signaling even under handoff among domains, even in movement among domains like movement within the same domain.

4 Security Analysis

Before explaining security analysis, several considerations are added as follows. (1) Although it was defined that the group key GK is safely shared between

LAAA and MAGs in advance, if an attacker has enough time and high-speed computer then, the key which is used for a long cycle can be under brute force attack for a long time. Therefore, the length of key is assumed to be long enough so that the system can endure strongly. Further, the system should change key timely to reduce the probability of hacking due to brute force attacks. (2) Security characteristics of SK-L²AM are based on one-way hash function (for example SHA-512 [25]) for collision avoidance. If χ value is given for one-way hash function $H()$ then, $H(\chi)$ is easy to calculate. But, if $H(\chi)$ is not given then, it is very difficult or it incurs high calculation costs to calculate this. Besides, SK-L²AM satisfies security characteristics as follows.

1. Replay Attack Resistance: It is difficult for the attackers to guess the value of the random number as the random numbers are newly created in each authentication procedure. As the random numbers are included in the authentication information (M_n) in order to prevent replay attack resistance, the proposed SK-L²AM has resistance in replay attacks.
2. Server Spoofing Attack Resistance: MR authenticates the authentication server and vice versa in SK-L²AM. This mutual authentication is ineffective spoofing attacks completely.
3. There is no time synchronization: To cope with replay attack, several authentication systems use time stamp mechanisms. But, time stamp mechanism may have several disadvantages such as other time zones and long transmission latency. But this system is the random number based authentication system. Therefore, this system does not have time synchronization problem.
4. Stolen-verified Attack Resistance: In SK-L²AM, AAA server does not have to save any verified information. Even if any attacker infiltrates in to the database of AAA server, he may not acquire any user authentication information. Therefore, SK-L²AM is strong to attacks for Stolen-verified Attack.
5. Message Modification Attack Resistance: To create message digest, one-way hash function is used safely so that information is not modified. If any attacker transmits packets which are modified (malignant) to MR or the authentication server then, the packets can be easily checked as the hash value is checked.
6. Local authentication: Local authentication has 3 advantages. Local authentication reduces satisfies authentication time. It also reduces satisfies network burden. And it provides fault tolerance mechanism. In other words, even if AAA server is in hacking, MR still carries out authentication procedure in domains.
7. Generation of session key: To provide safe communication in the first and fast re-authentication procedures of SK-L²AM, the session key which uses the random numbers is generated. In AX-FPMIPv6, the key is created in pre-handoff procedure. To complete the procedure, the MR and MAG can mutually communicate safely. Specifically, MR and MAG can use the session key to encrypt messages in order to prevent overhearing of their contents.
8. Known-plaintext Attack Resistance: Known-plaintext attack resistance is a cryptanalytic attack in which the attacker obtains both the plaintext and its

corresponding cipher text, and then the attacker tries to discover secret information. Although MAC_{MR} is transmitted in this system, it does not suffer from known-plaintext attack. The reason for not being able to attack using plaintext is because the attacker can obtain only MAC_{MR} but the attacker does not know the applicable secret key G (namely $G=H(x \parallel MAC_{MR})$) and secret value χ . Therefore, the attacker will know it is difficult to easily execute plaintext attack. In fast re-authentication procedure, we do not suffer known-plaintext attack owing to the same reason (the attacker can acquire only MAC_{MR} but does not know the applicable secret key K (i.e., $K=H(x \parallel MAC_{MR})$) and group key GK . Fast re-authentication procedure can also still resist in attacks. Because the attacker does not the applicable secret key K (i.e., $K=H(GK \parallel MAC_{MR})$) and the group key GK .

5 Performance Analysis

5.1 Evaluation Criteria

The proposed mechanisms based on the following performances evaluation criteria would be analyzed in three points of views as follows.

- Calculation Cost (CC): Complexity of the mobile node.
- Authentication Latency (AL): Latency between an authentication request sent by the MR and receiving of an applicable authentication reply.
- Handoff Latency (HL): Time when MR requires changing of the MR connection. The total handoff latency is the sum of L2 handoff latency, authentication latency and handoff latency in a network layer (L3).

A good authentication mechanism should incur low cost of calculation and provide low authentication latency. Moreover, low handoff is reinforced in the fast handoff and the integrated design of authentication and fast handoff should prevent overlapped signaling cost. Through analysis models and numerical results, our proposed mechanism is to show that it is a better solution for security and latency problems compared to the existing systems.

5.2 Parameter

Fig. 6 shows the network phases and numerical analysis model. Although signaling messages have different sizes, each signal message is assumed to have the same transmission latency and calculation costs. In the evaluation, notations as followings are used.

- D_{A-B} : Average delivery latency between node A and node B and it is presumed that $D_{A-B} = D_{B-A}$.
- D_{RA} : Time required to transmit the fast neighboring advertisement message.
- m : hop count between home and domains.
- $D_{PROC(A)}$: Average process latency of procedure A.

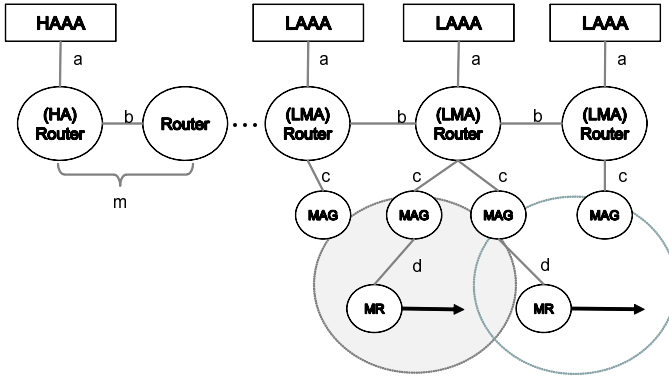


Fig. 6. Network phases and numerical models for performance evaluation

- Handoff latency can be expressed as the sum of L2 detection latency (D_{L2}), move detection latency (D_{MD}), overlapping address detection latency (D_{DAD}), authentication latency (D_{AUTH}) and location registration latency (D_{BU}). MAG which supports mobility so that router advertisement (RA) not request in [2] is sent more often suggests it should be established with more less MinRtrAdvInterval (MinInt) and MaxRtrAdvInterval (MaxInt) values. To simplify more, in the ground [26], it is assumed that it is half of the average value for RA message, which did not request the D_{MD} value (i.e., $(\text{MinInt}+\text{MaxInt})/2$), and the quarter of the average RA message value, which did not request the D_{MD} value in HMIPv6 (i.e., $(\text{MinInt}+\text{MaxInt})/4$).
- SA: The number of signaling messages which node A sent.

Table 2 is showing that values of parameters used in numerical analysis based on [26] and default latency value of DAD operation is 1000ms.

Table 2. Parameters used in the numerical analysis

Item	D_{L2}	D_{DAD}	a	b	c	d	MinInt	MaxInt	$D_{PROC(AUTH)}$
Time(ms)	50	1000	10	10	10	100	30	7	10

5.3 Analysis Results

Calculation Cost (CC) In this section, calculations costs of SK-L²AM and LR-AKE systems are compared. The analysis of calculation costs analysis, notations as followings are used. "-" means there is no calculation costs. n is the income of MRs which AAA server handles. C_h represents the cost at which one-way hash function is executed. C_{sym} represents cost for symmetric encryption or decryption. C_{asym} represents cost for asymmetric encryption or decryption. As LR-AKE does not support local authentication, the authentication procedure is

Table 3. Calculation Costs of SK-L²AM System

<i>Item</i>	<i>MR</i>	<i>HAAA</i>	<i>LAAA</i>	<i>MAG</i>
Home regist. proc.	-	nC_h	-	-
First auth. proc.	C_{ram}	-	$C_{ram} + 3C_{sym} + 5C_h$	-
Re-auth. proc.	$C_{ram} + 3C_{sym} + 3C_h$	-	-	$C_{ram} + 3C_{sym} + 4C_h$

performed every time in HAAA. Therefore, under LR-AKE system the bottleneck problem can arise in HAAA. Table 3 represents the calculation complexity of each SK-L²AM.

Authentication Latency (AL) SK-L²AM performances is evaluated by numerical analysis. It is compared with LMAM which is combined with the AAA system, the simple NEMO protocol, LR-AKE system and the system of Shi et al. Also, authentication latency is considered in three mobility scenarios. (a) When MR enters into a domain first (b) MR moves within the same domain (c) Last, MR moves among the domains. The numerical analysis of authentication latency is as follows. Table 4 is the result of the numerical analysis for authentication latency.

Table 4. Comparison of Authentication Latency

<i>Item</i>	<i>Domain initially</i>	<i>Same domain</i>	<i>Another domain</i>
Proposed	$2a + 2c + 2d$	$(MinInt + MaxInt)/2$	$(MinInt + MaxInt)/2$
LMAM	$2a + 2c + 2d$	$2d$	$2(2a + 2c + 2d)$
NEMO+AAA	$4a + 2mb + 2c + 2d$	$4a + 2mb + 2c + 2d$	$2(4a + 2mb + 2c + 2d)$
LR-AKE	$10a + 9mb + 5c + 5d$	$10a + 9mb + 5c + 5d$	$2(10a + 9mb + 5c + 5d)$
Shi et al.	$4a + 2mb + 2c + 2d$	$2a + 2c + 2d$	$2(4a + 2mb + 2c + 2d)$

When MR moves into another domain, the authentication latency is defined in Fig. 7 as follows. SK-L²AM show the way or means to achieve the least authentication latency out of the other compared approaches. This is the reason for using local authentication instead of home authentication. While the local authentication system shows similar results with the LMAM system, on the contrary LR-AKE system consumes long time on negotiation between HAAA and LAAA server. If the hop count m is large then the authentication latency would get more longer. When MR entered into a domain first, the system of Shi et al. should also send authentication information to HAAA server.

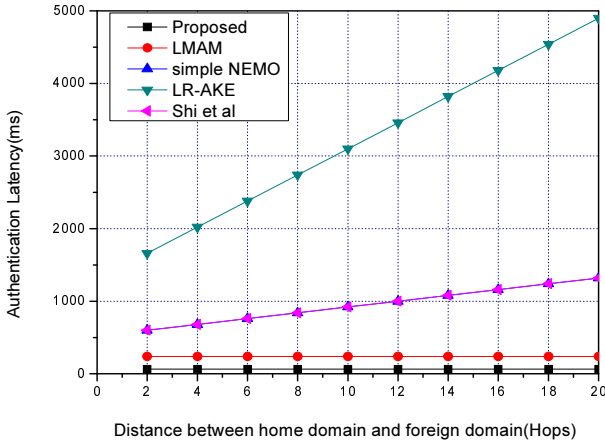


Fig. 7. Authentication latency when moving into another domain

Handoff Latency (HL) After simulating AX-FPMIPv6 performances, it compares with LE-HMIPv6 system, Simple NEMO system, LLH system and HMIPv6 system. Result was acquired as the average of 10 times in this simulation. HMIPv6 is assumed to support local registration. Total handoff latency is the sum of L2 handoff latency, authentication latency and L3 handoff latency. As SK-L²AM provides local authentication without sending information to HAAA server of MR, it operates effectively in mobility management within the local domains. Additionally, AX-FPMIPv6 uses Piggybacks method in order to reduce signaling overload and it also uses several L2 trigger to support prior handoff procedure. Therefore, handoff procedure among domains can be executed rapidly. Fig. 8 shows average handoff latency within the same domain.

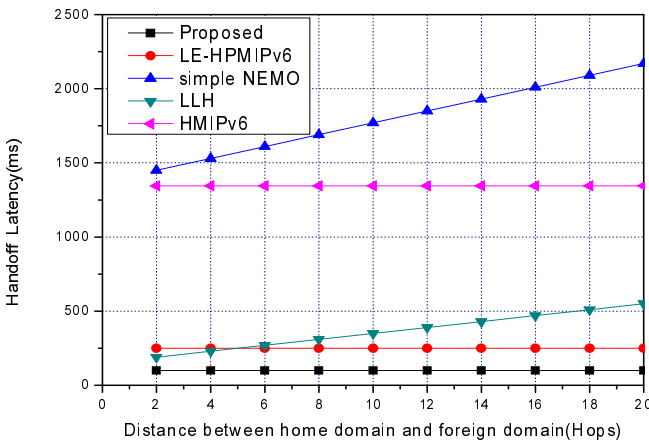


Fig. 8. Distance between home domain and foreign domain vs Average handoff latency within the same domain

Fig. 8 shows the results of handoff latency among domains at different distances (i.e., hop count) between home and foreign domains. As LE-HPMIPv6 supports mobility management and local registration systems but the application of HMIPv6 protocol improved the handoff from MIPv6. Handoff latency reduces but it causes the longest handoff latency for movement detection and DAD latency also it, does not support handoff among domains. A simple NEMO protocol inherited the disadvantage of long handoff latency in MIPv6 and it does not support local authentication. Although LLH uses prior registration method to reduce handoff latency, the authentication procedure should be executed in the HAAA server and due to this authentication latency arises. As our proposed system completes network-based architecture, local-based authentication, authentication and movement detection procedure in the pre-handoff stage, AX-FPMIPv6 mechanism have the lowest handoff latency. Further, to avoid Ping-Pong effect and the occurrence of HER problem, this paper uses the concept of "DeuceScan" and L2 triggers start the pre-handoff procedure in time.

6 Conclusions

As the local authentication mechanism called as SK-L²AM is proposed to support network mobility in this paper. Calculation cost related with code is considerably reduced for this use in the symmetric encryption and hash function. In order to also reduce the authentication latency, the authentication procedure can be completed without returning to HAAA or LAAA server. And PMIPv6 is improved to support local handoff and global handoff in a wireless network. X-FPMIPv6 uses many triggers and many CoAs to increase handoff procedure speed and avoid HER problem.

Lastly, as SK-L²AM supports local authentication, it does not increase the signaling overload. So, it is integrated in AX-FPMIPv6. According to the results of the performances analysis, it is shown to be more excellent than all the existing systems in calculation costs, authentication latency, handoff latency and signaling cost. Related to the security problem, SK-L²AM is very effective in local authentication, replay attack resistance, stolen verifier attack resistance, session key creation, mutual authentication to prevent server spoofing attack, known plaintext attack resistance and message alteration attack resistance.

Acknowledgments. This research was supported by Next-Generation Information Computing Development Program (No.2011-0020523) and Basic Science Research Program (2011-0027030) through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology. Corresponding author: Jongpil Jeong and Jae-Young Choi.

References

1. Devarapalli, V., Wakikawa, R., Petrescu, A., Thubert, P.: Network Mobility (NEMO) Basic Support Protocol. IETF, RFC 3963 (January 2005)
2. Johnson, D., Perkins, C., Arkko, J.: Mobility support in IPv6. IETF, RFC 3775 (June 2004)

3. Lamport, L.: Password authentication with insecure communication. *Communications of the ACM* 24(11), 770–772 (1981)
4. Narten, T., Nordmark, E., Simpson, W.: Neighbor discovery for IP version 6 (IPv6). RFC 2461 (December 1998)
5. Park, J.H., Jin, Q.: Effective session key distribution for secure fast handover in mobile networks. *Telecommunication Systems* 44(1-2), 97–107 (2009)
6. Mishra, A., Shin, M.H., Petroni, N.L., Clancy, J.T., Arbauch, W.A.: Proactive key distribution using neighbor graphs. *IEEE Wireless Communications* 11(1), 26–36 (2004)
7. de Laat, C., Gross, G., Gommans, L., Vollbrecht, J., Spence, D.: GenericAAA architecture. IETF RFC 2903 (August 2000)
8. Glass, S., Hiller, T., Jacobs, S., Perkins, C.: Mobile IP authentication, authorization, and accounting requirements. IETF RFC 2977 (October 2000)
9. Perkins, C.E.: Mobile IP joins forces with AAA. IEEE RFC 2976 (August 2000)
10. Calhoun, P., Johansson, T., Perkins, C., Hiller, T.: Diameter Mobile IPv4 application. IEEE RFC4004 (August 2005)
11. Fathi, H., Shin, S., Kobara, K., Chakraborty, S., Imai, H., Prasad, R.: LRAKE-based AAA for network mobility (NEMO) over wireless links. *IEEE Journal on Selected Areas in Communications (JSAC)* 24(9), 1725–1737 (2006)
12. Hideki, I., Seonghan, S., Kanukuni, K.: Introduction to Leakage-Resilient Authenticated Key Exchanged Protocols and Their Applications. In: *KHISC* (December 2008)
13. Wang, Y., Luo, W., Shen, C.: Analysis on Imai-Shin’s LR-AKE Protocol for Wireless Network Security. In: Bond, P. (ed.) *Communications and Networking in China*. CCIS, vol. 26, pp. 84–89. Springer, Heidelberg (2009)
14. Chubng, M.-C., Lee, J.-F.: A lightweight mutual authentication mechanism for network mobility in IEEE 802.16e wireless networks. *Computer Networks* (June 2011)
15. Li, D., Sampalli, S.: An efficient contributory group rekeying scheme based on hash functions for MANETs. In: *IFIP International Conference on Network and Parallel Computing Workshops*, pp. 191–198 (September 2007)
16. Ng, W.H.D., Sun, Z., Cruickshank, H.: Group key management with network mobility. In: *13th IEEE International Conference on Networks (ICON)*, vol. 2, pp. 716–721 (November 2005)
17. Kim, Y., Perrig, A., Tfsudik, G.: Group key agreement efficient in communication. *IEEE Transactions on Computers* 53(7), 905–921 (2004)
18. El Malki, K. (ed.): Low-Latency Handoffs in Mobile IPv4. IETF RFC 4881 (June 2007)
19. Koodli, R. (ed.): Fast Handoffs for Mobile IPv6. IETF, RFC 4068(June 2005)
20. McCann, P.: Mobile IPv6 fast handoffs for 802.11 Networks. IETF RFC 4260 (November 2005)
21. Gundaveli, S., Leung, K., Devarapali, V., Chowdhury, K., Patil, B.: Proxy Mobile IPv6. IETF RFC 5213 (August 2008)
22. Lee, K.-H., Lee, H.-W., Ryu, W., Han, Y.-H.: A scalable network-based mobility management framework in heterogeneous IP-based networks. *Telecommunication Systems* (June 2011)

23. Na, J.-H., Park, S., Moon, J.-M., Lee, S., Lee, E., Kim, S.-H.: Roaming Mechanism between PMIPv6 Domain (July 2008), [draft-park-netmm-pmipv6-roaming-01.txt](#)
24. Chen, Y.-S., Chuang, M.-C., Chen, C.-K.: DeuceScan:deuce-based fast handoff scheme in IEEE 802.11 wireless networks. In: IEEE Transaction on Vehicular Technology Conference, vol. 57(2), pp. 1126–1141 (September 2008)
25. NIST, U.S. Department of Commerce. Secure Hash Standard. U.S.Federal Information Processing Standard (FIPS) (August 2002)
26. Thomson, S., Narten, T.: IPv6 stateless address autoconfiguration. IETF RFC 2462 (December 1998)

An Architecture for Service Integration and Unified Communication in Mobile Computing

Ricardo Aparecido Perez de Almeida and Hélio Crestana Guardia

Department of Computer Science,
Federal University of São Carlos, Brazil
{ricardoalmeida,helio}@dc.ufscar.br

Abstract. Mobile communication devices often present different wireless network interfaces which allow users to access networked services from anywhere and at any time. Existing services, however, commonly require specific interfaces and infrastructure to operate, and often fail to convey different data flows to a device. Little context information is used in adapting contents to differing devices. This work presents a new context aware architecture for using the Bluetooth interface as a unifying communication channel with mobile devices. Content adaptation is provided as well as a mechanism for offloading services to the support infrastructure. The obtained results show the developed communication model is viable and effective.

1 Introduction

The advent of mobile computing has caused a revolution in personal computing, and in how users access their devices and data at any time and place. Different wireless technologies, such as GSM, UMTS, Wi-Fi and Bluetooth, allow data exchanges and the access to the Internet from mobile devices. The increasing adoption of mobile phones, smartphones, and tablets however is in a significant extent due to the new services which are made available to their users. Besides providing access to web pages, services range from the direct communication among users to specific entertainment, education, and business applications. All add value to the devices and make them almost indispensables to many of our daily activities.

News and e-mail reading, mobile learning, u-commerce, mobile marketing, and social networking interactions are examples of networked services, all of which commonly involve the transmission of multimedia contents. Framing such contents to each possible receiving device and adapting the transmissions to changing network conditions or to the choice of technology available is usually performed on a per-service basis. Several services just fail in attending these requirements and subject the users to non-ideal transmissions and poor media presentations. From the user's perspective, integrating the access to several services into a single interface, with reduced human interactions for service activation, is desirable. Having appropriate context information used for the adaptation of incoming contents to the characteristics of a device and to the user's preferences would also be convenient.

Considering the significant reduced processing capacity and mainly the power constraints of mobile devices, the possibility of offloading applications and activities to external devices is also relevant.

This work presents a context aware communication architecture aimed at providing a single service point to mobile devices using wireless interfaces, mainly the Bluetooth adapter. User preferences and different context information are used in the adaptation of media information directed at the mobile device. A service offloading mechanism based on web services is provided to allow users to access different types of services selected by them in a personal profile. Analysis about content delivery and encounters with mobile devices are also conducted to show the feasibility of the proposed architecture.

The remaining of this work is organized as follows: section 2 presents relevant related work; section 3 presents a quick overview about Bluetooth technology and some results related to interactions with mobile devices with Bluetooth enabled; section 4 presents a deep view of the developed communication architecture, focusing on its characteristics and operation. In section 5 we show the details of a reference implementation of the architecture and some obtained results, and, finally, section 6 brings the conclusions.

2 Related Work

Besides providing a basic mechanism for the interconnection of devices and peripherals, the Bluetooth layered architecture allows different protocols to be used for general purpose communications. TCP/IP and the RFCOMM protocol stacks, for instance, allow different types of services to be provided over Bluetooth.

The use of a Bluetooth channel from mobile devices in an e-commerce scenario was explored by the Model for Ubiquitous Commerce Support – MUCS [1]. Using this system, buyers interests were automatically matched to the offers made available by the sellers in the same shopping region. Communications occurred using a wireless interface available to identify the participant devices which run the same specific application. MUCS was focused on its single purpose and did not consider security and content adaptation aspects in the data transmissions.

The Bluespots [2] project intends to act as a transmission channel placed into buses to create a delay tolerant network infrastructure. The idea is to provide passengers with access to specific services, such as e-mail, Web navigation, and file transfer from their mobile devices using Wi-Fi and Bluetooth interfaces. Each Bluespot includes a Wi-Fi access point and a Bluetooth device acting as the master of a piconet, which forwards user data to and from a repository called Data Store. Each repository acts as an intermediary cache holding web pages, news and multimedia contents obtained from eventual interactions with fixed Wi-Fi spots present at the bus stations. E-mails received from the users may also be forwarded through the Wi-Fi spots for latter deliver to the Internet. No notice could be found on the effective implementation of the proposed approach or on the details related to the envisioned data forwarding schemes. Other aspects related to security and content adaptation could not be determined.

iGrocer [3] integrates the use of user profiles and smart-phones for selecting purchase items in a store. Information provided by the user is kept in a profile regarding a list of products of interest and nutrition restrictions. A buying list is defined by the user from what is available in a particular store and which satisfy the restrictions previously defined. Once a list is concluded, a route map to pick up the selected items can be generated and the payment can be automated using a credit card. Help can also be provided to the user in finding the selected products at the store. Security concerns are taken into account in the communications.

The P-Mall [4] project presents a context-aware communication architecture for pervasive commerce. Users running a specific application on their mobile communication devices while walking in a specific shopping area can receive information about products of interest and suggestions about nearby shops offering related deals. Detailed information about a product, identified by its RFID tag, can also be fetched automatically from the Internet and presented in the user's device.

OOKL [5] allows mobile devices to be used to present or capture multimedia information about nearby places and objects of interest. A web site is used to centralize the information available. Besides receiving complementary information users are encouraged to do further research and to contribute contents to the database. The efficiency of this approach is significant [6], but there are restricting factors related to the use of 3G communication systems. Content adaptation is sought by creating versions of the program to each supported device.

Despite the existence of several mechanisms to provide services to mobile devices, most of the implementations presented are focused only on specific purposes, do not consider content adaptation and context-aware aspects, or are dependent on an always on communication system, such as the 3G infrastructure. The communication architecture presented here was developed to address these aspects.

3 Bluetooth

The Bluetooth technology [7] was created to provide wireless communication between mobile and stationary devices. Operating over the ISM (Industrial, Scientific and Medical) radio frequency band, this technology is available at a range of different types of mobile devices, such as notebooks, netbooks, tablets, MP3 players, PDAs, cell phones, smartphones, etc. Different versions of Bluetooth technology have been developed since its creation (1.2, 2.0, 2.1, 3.0 and 4.0) and the standard transmission rates have increased from 1 Mbps to 3 Mbps, in versions 2.0 and 2.1, and up to 24 Mbps in versions 3.0 and 4.0. Most of the mobile devices with Bluetooth technology available nowadays support the 2.0 or 2.1 versions.

Considering just cell phones and smartphones, it is estimated that there will be about 1 billion devices with Bluetooth by 2014 [8]. The widespread presence of this technology in mobile devices makes it a promising communication channel. The modular architecture of its implementation also allows an unlimited number of communication services to operate on top of it. However, using Bluetooth to provide

services to mobile devices depends on understanding how the data transmission occurs and how users interact with other devices using this technology.

Using the discovery service of the Bluetooth protocol, we have investigated several aspects regarding the adoption of this technology and user reachability for short range transmission. The results refer to data collected during a period of 10 months in a Computer Science department. Although there was a visible computer screen displaying information about the project, some of the users may not have been aware of the data collection experiment. No data from the experiments are made public or provided to other users. The performed analysis is related to the bulk data.

The obtained data provides an approximate idea on the adoption of different types of devices, and on how frequently users are seen using their devices in a discoverable mode. The referred university department has about 600 fixed students, considering both undergrads, and graduates. A total of 644 different devices were seen by the experiment, generating about 40.000 encounter records.

The results are presented in Figures 1, 2 and 3, and provide an idea of how widespread the use of such devices is nowadays. Although not shown here, it was possible to identify an increase in the adoption of devices which support a richer set of functionalities, mostly smartphones.

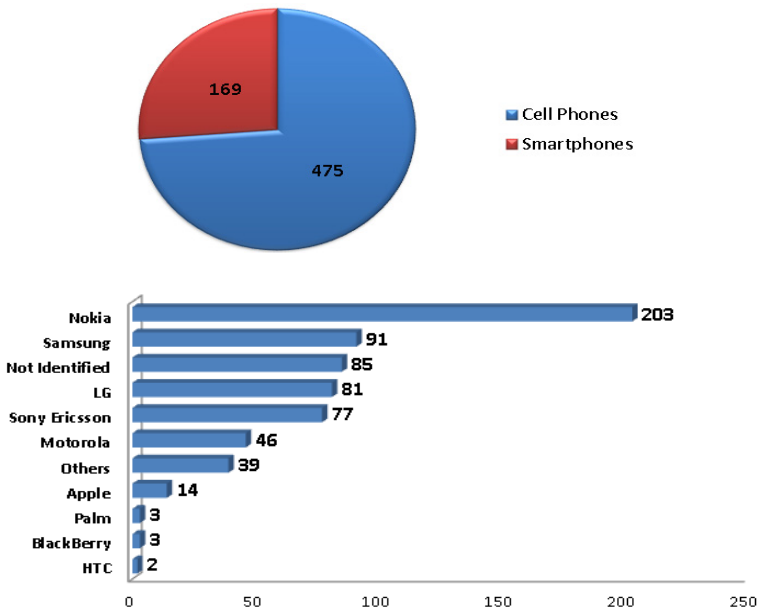


Fig. 1. Number of different cell phones and smartphones detected and number of devices detected by brand

Figure 1 shows that, based on the major and minor numbers that are transmitted by the Bluetooth adapter during an Inquiry operation, approximately 74% of the detected devices were identified as cell phones and 26% were smartphones. Using part of the Bluetooth MAC Address number of a detected device, in most cases it was also possible to identify the corresponding manufacturer.

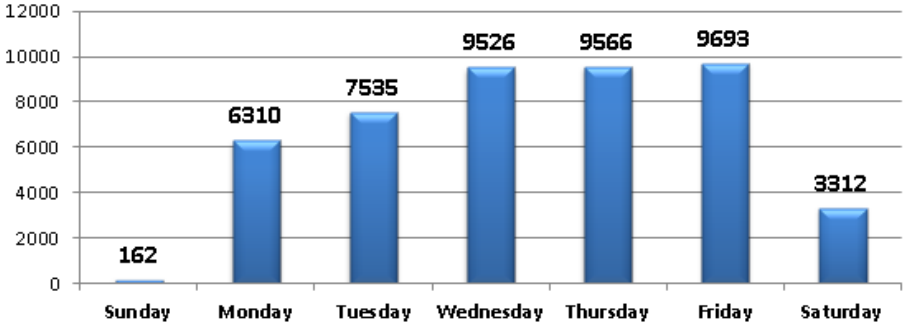


Fig. 2. Number of encounters by day of the week

Figure 2 shows that most of the encounters occurred on Wednesdays, Thursdays and Fridays, which are the days that concentrate most of the academic activities in the department. It is also possible to see on Figure 3 that most of the encounters occur during the time corresponding to the interval between classes, when the students are mostly nearby the monitoring point.

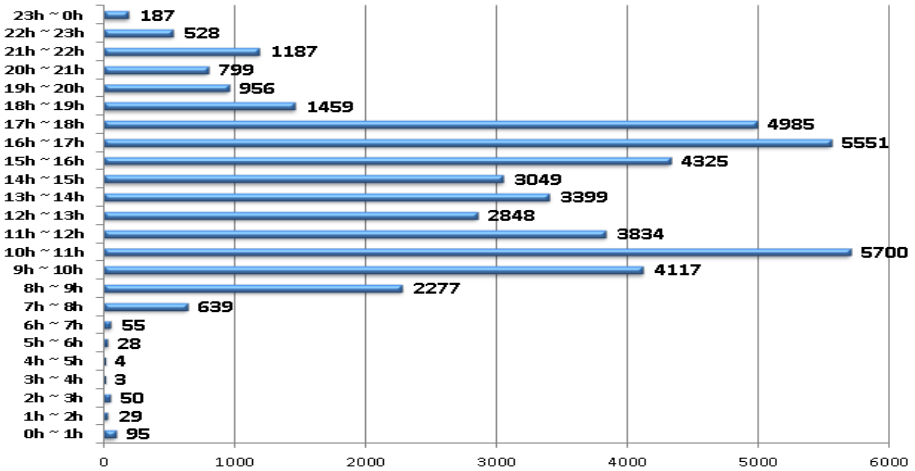


Fig. 3. Encounter dispersion along the day

Figure 4 presents relevant information about the duration of the encounters, which indicates the time a user’s device is visible by the monitoring point after it is identified. The monitoring point was situated in the hallway of the Department, which is a place most people just pass by. Using a total of 32.609 complete records from the dataset encounters were seen to vary from just a few seconds to up to 45 seconds.

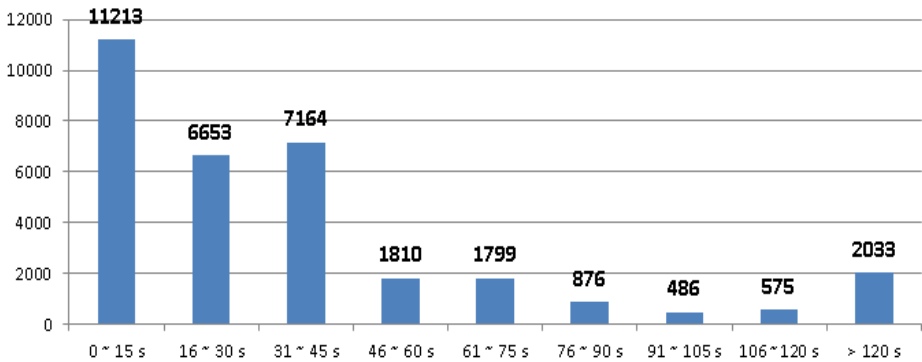


Fig. 4. Time encounter duration

From the initial data set it is possible to note that the Bluetooth technology is in frequent use by a significant number of people, even though the data in this case may be biased to age and to a natural interest for technology within this group.

4 Architecture Description

The Bluetooth layered architecture includes protocols and services and different devices provide different functionalities on top of it. The plethora of services which run on Bluetooth varies enormously from the simplest mobile phones to the current smartphones, and this is a concern if one wishes to provide data directly to the devices. Content adaptation to the varying hardware and Operating System platforms is also a concern, especially in the case of formatting different media files to enable them to be displayed according to the display resolution of a device.

Given the varying, usually reduced, processing capacity in the set of currently deployed mobile devices, service offloading is particularly important. This means the ability to transfer the execution of different applications from the mobile devices to servers present in the fixed infrastructure which just sends the results to the devices. A unified mechanism should be provided for this task. In the same way, the existence of a uniform mechanism for forwarding different data contents to the devices would be valuable.

In this sense, this work presents a communication architecture, called BlueYou, which was designed to provide a comprehensive bidirectional channel by which applications running in mobile devices can interact with an external service infrastructure. The developed architecture also allows different incoming data flows to a device to be properly adapted according to the characteristics of this device or to a user's preferences. The communication services envisioned that can benefit from the developed architecture are parts of a distributed environment with communications occurring via wireless transmissions interfaces.

The main elements that compose the architecture are presented in Figure 5.

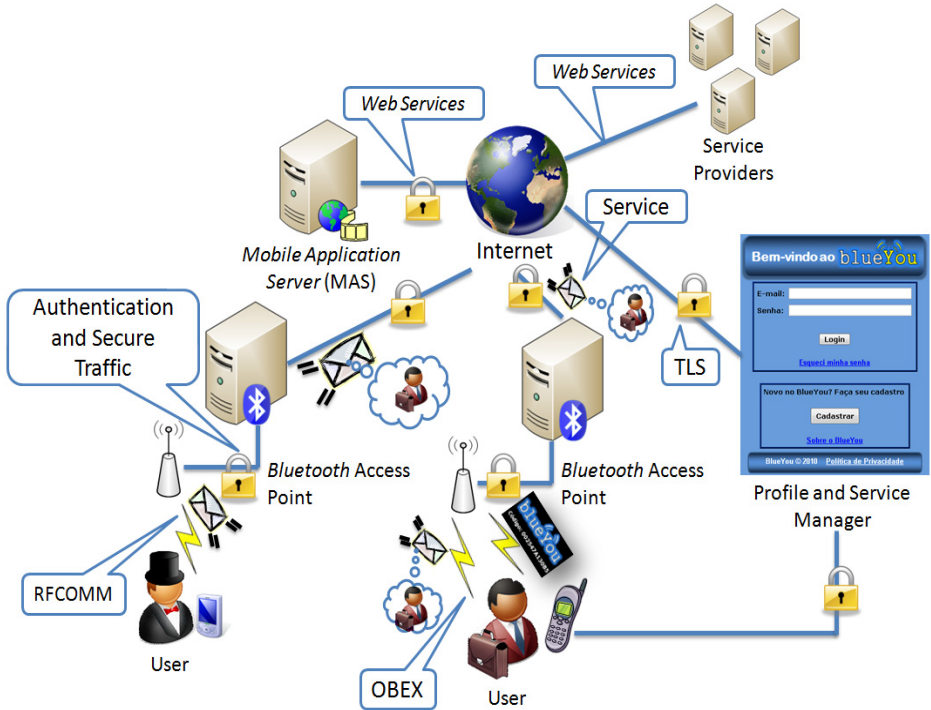


Fig. 5. Main elements of the BlueYou architecture

Each user is associated with a mobile device, which contains a unique Bluetooth MAC address. Information about the users, their corresponding devices, and their preferences are kept in a database of user profiles. This database includes information about the users (restricted to contact information), and about communication services to be performed by the BueYou architecture to each user.

The management of a user's profile information can be performed either using a standard web page or by a desktop application. A web service interface is also provided for direct calls to the profile keeper node. Using the predefined web page a user can select from a list of registered (known) services those that will be delivered to her mobile device. New processing and communication services can be registered in the application service using a standardized communication interface.

A PIN number can be informed by the user that will allow future communications from the infrastructure with the corresponding device to use the OBEX protocol and the Object Push [9] profile for the authentication of the communication entities.

A Mobile Application Server (MAS), as defined in [10], is also present in the proposed architecture. It is responsible for identifying the users nearby the access point, for defining the services that should be provided to those users and for adapting any

contents being transmitted, according to the context information available, prior to their effective delivery from a Bluetooth access point near a device.

The MAS is also responsible for intermediating the communications with any service providers that should be contacted for the services selected by a user in her profile. Web services are used for the interactions between the MAS and the original service providers, while TLS (Transport Layer Security) is used between the BlueYou components. A standardized interface is used to provide access to any service provided by the architecture. Other services can be easily added using the same interface, which allows the transmission of many content types (e.g. audio, video, image or text) to the mobile devices.

Bluetooth access points are used for the effective delivery of data to a user's device. Each access point must then be connected to the Internet or have some transmission infrastructure that allows the interaction with the MAS using web services.

Bluetooth Inquires are performed constantly at every access point. Every time a device is identified via a Bluetooth Inquiry procedure, the corresponding access point will contact the MAS to determine if the newly found device is associated with a registered user. If a current system user is identified, any pending information associated with the selected services is forwarded to her. If information about the device is not found by the MAS, this means a new user is met. Data about the encounter just occurred is kept in the system log and is not subject to disclosure.

The access points may be configured to approach new users just met. Upon request from the MAS, a new user is approached by sending an image file to her device using the OBEX protocol. The transmitted file contains a unique code associated to the device's Bluetooth MAC address. If the user accepts to receive the object being sent, she can later use the information contained in the file to register as a new BlueYou user and configure a profile for the corresponding device. The association of information about a registered device and the corresponding service profile and the encounters occurred is obtained from matching the MAC addresses.

Registered users are eligible to receive the contents associated to the selected services every time they approach a BlueYou access point with their devices configured with the Bluetooth interface active and in the discoverable mode. Transmissions from the access points to the mobile devices can occur with no need to run specific application codes on the devices. This is achieved by using the OBEX protocol and the Object Push profile, commonly active in most devices with Bluetooth interfaces. On the other hand, extended transmission functionalities are provided if a specific program is running on the mobile device. The RFCOMM [11] protocol is used in this case and allows a mobile device to initiate a communication with a BlueYou access point. An infinite number of applications can be implemented to benefit from the communication capability provided by this protocol.

Interactions may also occur as a result from a mobile device's application initiative. In that case, data generated at the device can be forwarded to a BlueYou access point. The required communications initiate with an authentication phase during which the user must provide the appropriate PIN number previously registered in her profile. Once authenticated, data transmissions may occur from the device to the access point using encryption for security reasons. Using the RFCOMM protocol it is

also possible to automate the transmissions for data forwarding within a BlueYou enabled application with no need for user intervention.

When sending information to a device, context information, as well as data from the user's device profile, are taken into account for content adaptation. This includes information about the size of the screen, the file types supported, network adapters present in the device, transmission rates supported, etc. Some information about the user's preferences is also considered in the content adaptation, such as the choice for graphical or textual contents, default font size, etc. Information about the user's current location and time of day are also considered in determining which services should be made available. Different information and forms of content adaptation can also easily be introduced in the system.

5 Implementation

A reference JAVA implementation of the BlueYou architecture was created for validation purposes and was installed at the Computer Science Department of the Federal University of São Carlos. The MAS role in BlueYou is implemented as a set of web services, which are used to receive update information sent from the fixed active access points about the registered devices/users currently seen. It is the role of the MAS service to also forward any pending data aimed at these users to the corresponding access points.

Services in the BlueYou architecture are implemented as web services and are accessed using the REST (REpresentational State Transfer) architectural style. Requests are sent using URLs and HTTP basic operations (GET, POST, PUT and DELETE). Responses are encapsulated in standardized JSON (JavaScript Object Notation) documents (one for services provided over OBEX and another for services provided over RFCOMM). An authentication mechanism is used for every request to the MAS, in order to avoid not authorized accesses.

Users can create and maintain a BlueYou profile using a web browser to access the JSF (JavarServer Faces) web page developed for this project, as shown in Figure 6. Personal information requested is kept at a minimum, and corresponds basically to a valid contact e-mail. Other information currently kept in a profile includes details about the user's mobile device, only used for content adaptation purposes.

A login name must be defined along an associated password. The access code received by the device in a previous approach from any BlueYou access point must also be provided. All the communication with the web page occurs using the TLS protocol.

All contact information about a mobile device is kept by the system and are used solely for profile studies intended to improve future interactions with the users. Any investigation concerning the gathered data shall be done in an obfuscated data set. As stated in the BlueYou Privacy Policy, a user can request that her current profile and all associated information kept be removed at any time.



Fig. 6. Current BlueYou user profile web page

Two JSE applications are run at a BlueYou access point. The first is used to forward the data associated with the user's selected services to their devices when they are near this access point. Once the service data are obtained from the MAS, they are forwarded to the devices using the OBEX protocol.

The second application is responsible for mediating the data transmissions initiated at the user devices running a specific JME application. Data transmissions in this case occur using the RFCOMM protocol.

Other applications can be developed to make use of the BlueYou API, which is also used in the implementation of the applications run at the access points. Using this API, a single logical interface, accessible via *send()* and *receive()* methods, provides transmissions and receptions over different protocols. OBEX and RFCOMM are currently supported but the standard socket mechanism could also be used underneath. The class diagram for the BlueYou API (BY) is presented in Figure 7.

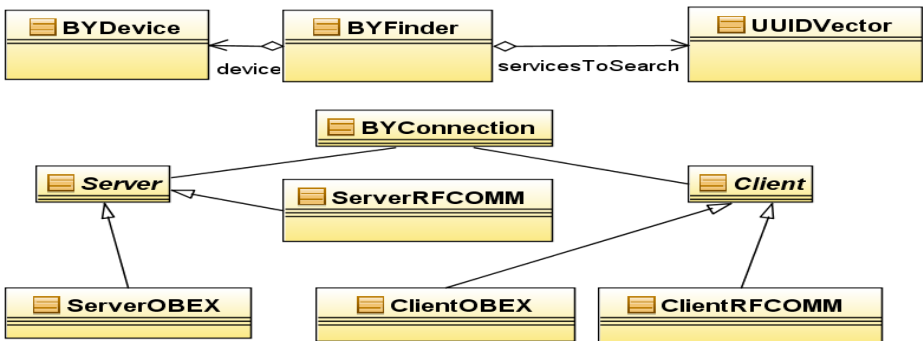


Fig. 7. Classes diagram for the BY API

The BYFinder class is used for searching other nearby BlueYou enabled devices. This is done by running Bluetooth inquiries using an object of the BYDevice class. A list of all devices and services supported is obtained in return, including any service associated with the standard Bluetooth profiles, such as Object Push. An object of the class UUIDVector is used in this case. The BYConnection class is used for establishing a connection using the OBEX or the RFCOMM protocols, and returns an object either of the Server or the Client classes. Both Server and Client possess abstract methods send() and receive() implemented in their inner subclasses.

The application code used at the access points (APs) performs periodic Bluetooth inquiries searching for discoverable devices in the neighborhood. When a device is found, the AP sends a query to the MAS about the identity of this device and, if it belongs to a registered user, which service data should be forwarded to it. If a non-registered device is found, the AP will also probe its list of active Bluetooth services and their corresponding connection strings. The obtained information is sent to the MAS, who keeps all received data in a MySQL database. Again, any information in this database is kept only for personalization purposes and is not subject to any form of unauthorized disclosure.

After the newly found device is registered, the MAS creates an image file containing a BlueYou code specific to this device. This file is then sent to the AP, which will try to forward it to the corresponding nearby device using the OBEX protocol. At this point, the user is automatically prompted by her device and may decide to accept the incoming file. Although the file name and the sending access point's name will be associated with the BlueYou project, there is currently no mechanism to prevent a spurious device from spoofing a valid BlueYou access point prior to accepting a transmission.

The file sent from the access point to the device is a harmless image file, which may promptly be shown to the user if the device's operating system has a default application associated with that file type. Using the code visible in the image file, the user may access the BlueYou web page and register for future interactions.

When a registered user is seen from a BlueYou access point (AP), the MAS sends back to this AP all pending data associated with the user's currently configured services. A content adaptation may happen at this point based on the device's characteristics. Content adaptation is performed by matching the device type and model provided by the user and databases containing information about known devices, such as the UAProf [12]. The effective data received by the access point is then forwarded to the appropriate device using the OBEX protocol.

5.1 Test Cases

A few sample services are currently in use at the experimental testbed, including the daily menu from the academic restaurant, the weather forecast for the city of São Carlos, access to tweets posted on Twitter and a user location system based on the users' contacts with the currently active access points. Figure 8 illustrates the weather forecast and menu services.



Quinta-Feira, 25/02

Boa tarde Ricardo,

Previsão do Tempo:

Cidade: São Carlos
 Nascer Sol: 06h06 **Por Sol:** 18h43
 Temp. Min: 21°C **Temp. Máx:** 26°C
 Prob. Chuva: 90% **Qtd. Chuva:** 15mn
 Vel. Vento: 17km/h

Restaurante Universitário

Almoço 

Prato principal: Frango Xadrez
Guarnição: Polenta
Arroz: Simples
Feijão: Simples
Saladas: Alface e Tomate
Sobremesa: Fruta
Bebida: Suco

Fig. 8. Weather forecast and daily menu services

Twitter feeds can also be retrieved as a BlueYou service. Any tweet posted on a Twitter blog followed by a user can be accessed using a BlueYou service. This is done by sending a text file to her device, containing the last posts from the followed blogs. An application was also developed to allow a BlueYou user to forward her tweets from a mobile device via a BlueYou access point. This application must be executed in the user's device, which will communicate via Bluetooth with a software running on the access points. The tweet is forwarded by the AP to the MAS, which will upload the post using the OAuth (Open Authorization) standard. Figure 9 provides an example text file containing the tweets received by a user, and the mobile application used for forwarding one's tweets.

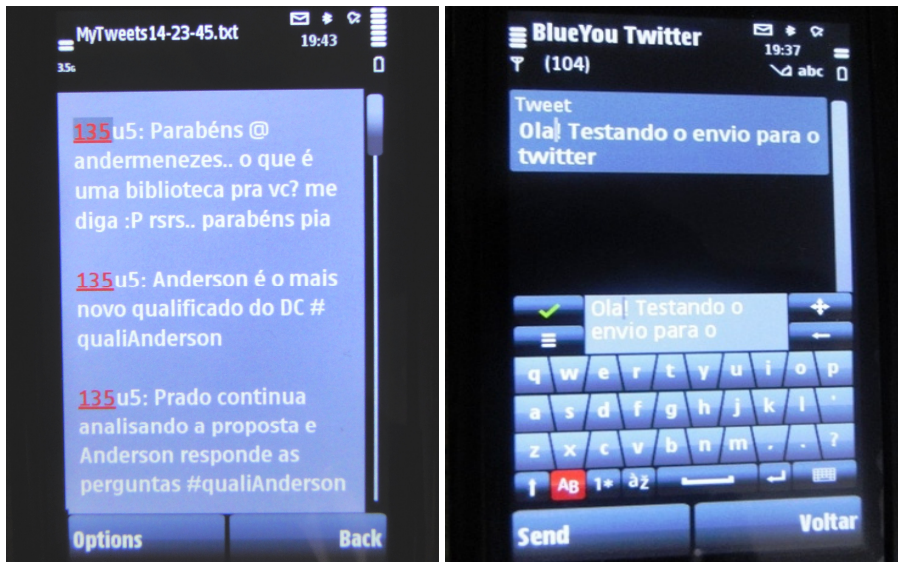


Fig. 9. Reception (left) and transmission (right) of Twitter messages using BlueYou

A user location service has also been implemented on top of the BlueYou architecture. This service allows an authorized user to find out about the location of another registered user according to where she was last seen near a BlueYou access point. Different location update mechanisms may also be incorporated to the system. A specific application must be run in the mobile device to receive and present an image obtained from the access point. This image is generated from a mashup using the GPS coordinates of an access point and a map from Google Maps. An example location image is shown in Figure 10.

A trusted relationship must be established in the user profile to create a list of other registered users allowed to make probes about her latest contact with a BlueYou access point.

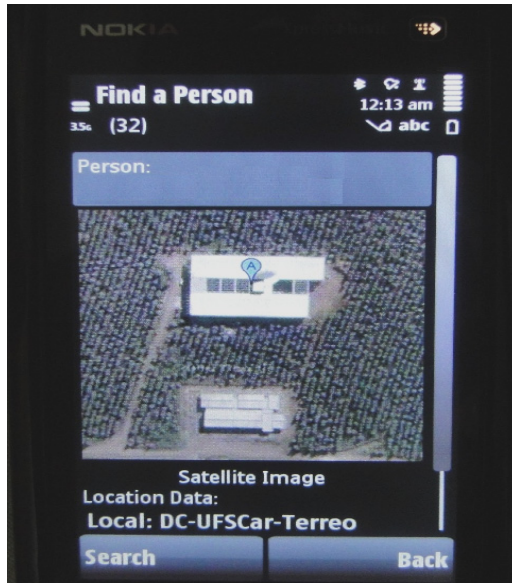


Fig. 10. BlueYou user location service, based on a mashup using Google Maps

5.2 Performance Results

An evaluation of the current implementation was performed, focusing both on the ability to interact with different devices and on determining the transmission characteristics for different content types. The tests correspond to 1200 transmission using 5 different mobile devices: Nokia N78 (Bluetooth version 2.0), Nokia 5800 Xpress Music (Bluetooth version 2.0), Motorola K1 (Bluetooth version 1.2), Samsung i5500 Galaxy 5 (Bluetooth version 2.1) and LG KS360 (Bluetooth version 2.0). The Bluetooth access point used was a Class 2 (max. range 20 meters) Bluetooth USB Dongle (version 2.0).

Four different contents were transmitted to each device 20 times: weather forecast (18 KB), academic restaurant menu (22 KB), text file with messages from Twitter (50

KB) and another text file with messages from Twitter (150 KB). In order to evaluate the effect of distance on the transmissions the tests were performed keeping the devices at 3 different positions relative to the access point, 3m, 6m, and 12 meters. The results are shown in tables 1, 2 and 3.

Table 1. Time Transmission (in seconds) – devices at 3m from the access point

Device Name	Weather Forecast (18KB)	Restaurant Menu (22KB)	Text Message (50KB)	Text Message (150KB)	Overall Transmission Rate (KB/s)
Nokia N78	1,68	1,55	2,33	5,68	18,20
Nokia 5800 Xpress Music	2,83	3,36	3,47	7,66	11,72
Motorola K1	10,60	12,62	37,93	38,25	2,17
Samsung i5500 Galaxy 5	1,46	1,93	2,74	6,65	16,13
LG KS360	2,85	2,94	4,63	7,86	10,92

Table 2. Time Transmission (in seconds) – devices at 6m from the access point

Device Name	Weather Forecast (18KB)	Restaurant Menu (22KB)	Text Message (50KB)	Text Message (150KB)	Overall Transmission Rate (KB/s)
Nokia N78	1,54	1,78	2,92	6,87	15,75
Nokia 5800 Xpress Music	3,72	3,80	3,73	7,90	10,75
Motorola K1	13,82	15,28	41,62	43,87	1,84
Samsung i5500 Galaxy 5	1,97	2,84	4,76	8,50	11,26
LG KS360	3,01	3,27	4,50	7,98	10,65

Table 3. Time Transmission (in seconds) – devices at 12m from the access point

Device Name	Weather Forecast (18KB)	Restaurant Menu (22KB)	Text Message (50KB)	Text Message (150KB)	Overall Transmission Rate (KB/s)
Nokia N78	1,71	2,22	2,73	6,81	15,19
Nokia 5800 Xpress Music	4,90	4,38	5,56	9,43	8,40
Motorola K1	16,61	18,56	43,63	45,81	1,67
Samsung i5500 Galaxy 5	2,19	2,84	4,50	8,61	11,12
LG KS360	3,06	3,00	4,60	7,78	10,84

The results show the influence of the Bluetooth version present in the devices. In most cases, it is possible to transmit to the devices at about 10 KB/s, as shown in the last column of the tables. For devices using the same version of the Bluetooth protocol, other aspects inherent to the chipset, hardware and operating system also seem to play a significant role. Given the low level of interference in the environment where

the tests were performed there was not a significant performance variation for the 3 distances evaluated.

Regarding the results obtained for the time devices are in contact when passing by an access point, most of the devices can be set to receive and transmit relevant information using the developed architecture. Comparing the results in tables 1 and 2 with those on Figure 4, less than 10 seconds is the time necessary for the typical file sizes tested, while most encounters last up to 45 seconds.

6 Conclusions and Future Work

Besides acting as an interconnection means between computer devices and different types of accessories, the Bluetooth technology provides a data path between these devices that can be used by high-level applications. Several different applications have been developed which explore this communication facility in a circumstantial way. The architecture presented in this work provides an abstract channel over the Bluetooth technology, which allows an unlimited number of applications to benefit from this communication model in a uniform way. The data transmission model provided by BlueYou is independent of the data transmission protocol used with the Bluetooth interface.

The architecture is deeply concerned about security and privacy. Any data transmitted from the services can be encrypted using digital envelopes, which also allow the nodes to authenticate every received information. Considering a scenario where a wide number of BlueYou access points are deployed, sensitive information about a user location can be gathered by the system. To avoid risking any of these data to be accidentally exposed, the access to the MAS server is exclusively based on a restricted set of web services and requires authentication before the communication. A privacy policy defined in the system also assures its commitment to not disclose any data about the registered encounters. Users may request that information associated with a device be removed from the system logs.

The set of applications currently implemented show the viability of the architectural model developed. Considering a user passing by a BlueYou access point equipped with a commodity Class 2 device, on average, less than 10 seconds are necessary for this access point to transmit the available services currently. The current version of the evaluation scenario has more than 100 users, which frequently access their services.

Scalability is also inherent to the distributed implementation model used in BlueYou, as new MAS servers can be added to support a group of access points.

References

1. Franco, L.K., et al.: Um modelo para Exploração de Oportunidades no Comércio Ubíquo. In: Proceedings of Latin American Informatics Conference, XXXV, Pelotas, RS, Brasil, pp. 1–10 (2009)

2. Lebrun, J., Chuah, C.-N.: Bluetooth Content Distribution Stations on Public Transit. In: Proceedings of International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking, Los Angeles, CA, pp. 63–65. ACM (2006), <http://doi.acm.org/10.1145/1161252.1161269>
3. Shekar, S., Nair, P., Helal, A.S.: iGrocer: a ubiquitous and pervasive smart grocery shopping system. In: Proceedings of ACM Symposium on Applied Computing, pp. 645–652. ACM, New York (2006), <http://doi.acm.org/10.1145/952532.952658>
4. Lin, K.-J., Yu, T., Shih, C.-Y.: The Design of A Personal and Intelligent Pervasive-Commerce System Architecture. In: Proceedings of 2nd IEEE International Workshop on Mobile Commerce and Services, Munich, Germany, pp. 163–173. IEEE Computer Society (2005), <http://dx.doi.org/10.1109/WMCS.2005.25>
5. Ookl: (December 2010), <http://www.ooklnet.com/web/whatisthis.php>
6. Salaman, A.: Hand-held Learning (December 2008), <http://ookl.files.wordpress.com/2008/10/nmm-gem-article.pdf>
7. Bluetooth Sig. Basics (December 2010), <http://bluetooth.com/Bluetooth/Technology/Basics.htm>
8. Mawston, N.: Enabling Technologies: Global Bluetooth Phone Sales by Bluetooth Profile (December 2009), <http://www.strategyanalytics.com/default.aspx?mod=ReportAbstractViewer&a0=4918>
9. Bluetooth Sig. Object Push Profile (December 2001), http://www.bluetooth.com/SiteCollectionDocuments/OPP_SPEC_V11.pdf
10. Jain, R., et al.: The Mobile Application Server (MAS): An Infrastructure Platform for Mobile Wireless Services. *Information Systems Frontiers* 6(1), 23–34 (2004), <http://dx.doi.org/10.1023/B:ISFI.0000015872.20136.76>
11. Bluetooth Sig. RFCOMM with TS 07.10. Bluetooth SIG (December 2003), <http://www.bluetooth.com/SiteCollectionDocuments/rfcomm1.pdf>
12. Alliance, O.M.: User Agent Profile (UAPProf) (December 2010), <http://www.openmobilealliance.org/Technical/Schemas.aspx>

Task Allocation in Mesh Structure: 2Side LeapFrog Algorithm and Q-Learning Based Algorithm

Iwona Pozniak-Koszalka, Wojciech Proma, Leszek Koszalka,
Maciej Pol, and Andrzej Kasprzak

Dept. of Systems and Computer Networks, Wrocław University of Technology,
Wrocław, Poland

`iwona.pozniak-koszalka@pwr.wroc.pl`

Abstract. This paper concerns the problem of task allocation on the mesh structure of processors. Two created algorithms: 2 Side LeapFrog and Q-learning Based Algorithm are presented. These algorithms are evaluated and compared to known task allocation algorithms. To measure the algorithms' efficiency we introduced our own evaluating function – the average network load. Finally, we implemented an experimentation system to test these algorithms on different sets of tasks to allocate. In this paper, there is a short analysis of series of experiments conducted on three different categories of task sets: small tasks, mixed tasks and large tasks. The results of investigations confirm that the created algorithms seem to be very promising.

Keywords: mesh structure, task allocation, algorithm, reinforcement learning, Q-learning.

1 Introduction

The popularity of mesh oriented systems is still increasing because of also still increasing demand for computers with higher computational power. Meshes are relatively simply and regular structures [1], therefore they are considered as one of the best solution for multiprocessor systems.

For all task allocation methods, the priority is to maximize as higher computational power as it is possible using available source [2]. In real cases, the full computational power is not achieved because of difficulty with the task allocation on parallel connected processors. In order to increase the efficiency of task allocation process - various ideas, approaches, and methods in constructing task allocation algorithms are used. In this paper, we propose two new task allocation algorithms, including the algorithm based on reinforcement learning idea [3].

The rest of the paper is organized as follows. In Section 2, the basic terminology and objective function is presented. Section 3 contains short description of known task allocation algorithms. In Section 4, the created algorithms are presented. Section 5 contains results of investigations, including a brief analysis of results of series of simulation experiments made for three categories of sets of tasks. The final remarks appear in Section 6.

2 Tasks Allocation

2.1 Terminology

The basic terms used in the task allocation problem can be defined as follows [4]:

- node – the basic element of mesh-oriented system describing one singular processor; one node has two possible states: free and busy;
- mesh – two-dimensional structure consisted of uniformly distributed nodes; one mesh is described as $M = (w, h)$, where w is the width and h is the height of mesh;
- submesh – group of nodes connected in rectangular structure which is a part of mesh M ; one submesh is described as $S = (x, y, x', y')$, where x and y are coordinates of the upper left corner of submesh, x' and y' are coordinates of the lower right corner of the submesh; the submesh is busy only if every nodes in submesh are busy;
- task – treated as group of nodes and described as $T = (w', h', t')$, where w' is the width, h' height and t' time duration of one task; tasks are grouped in task list; in the allocation process, tasks are being removed from the list and allocate on mesh creating busy submeshes;
- fragmentation – there are two types: internal and external; internal fragmentation is the process of rounding up the task area – in definition, it is a ratio between the surplus of unused nodes in submesh and the number of all busy nodes in submesh; external submesh is the ratio between the number of nodes from the space between tasks already allocated on mesh and all busy nodes in already allocated tasks (busy submeshes).

An example of the mesh for $M = (7, 5)$ is shown in Fig.1.

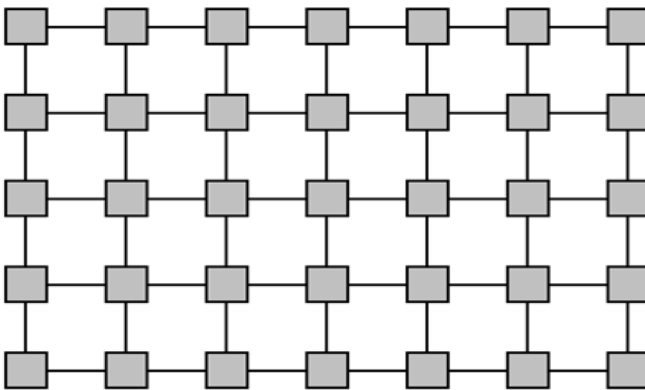


Fig. 1. Topology of mesh

2.2 Problem Statement

The considered allocation problem can be formulated as follows:

For given:

- mesh with given size $M = (w, h)$,
- list of tasks $T = T_1, T_2, T_3, \dots, T_n$ with known parameters (w'_i, h'_i, t'_i) .

To find:

the sequence of the tasks on the list of tasks.

Such that:

the desirable properties of the defined objective function are fulfilled.

2.3 Objective Functions

In order to compare the task allocation algorithms we need the measure of efficiency, i.e., the index of performance allowing to access the allocation process made using a given algorithm. Firstly, we introduce an auxiliary index of performance. The OB rate is the ratio between all busy nodes in the mesh and all nodes in the mesh. It may be calculated in the moment, when there is no possibility to allocate (by a given task allocation algorithm) a latter task (due to the full allocation by just allocated former tasks on the mesh) and the algorithm must wait until it occurs any free submesh acceptable by this algorithm. The OB can be expressed by the formula (1):

$$OB = \frac{\sum_{i=1}^n (w'_i * h'_i)}{w * h} \tag{1}$$

where n is the number of already allocated tasks on the mesh. In the numerator of (1) is the sum of nodes which are currently busy (w'_i and h'_i are the width and height of submesh i). In the denominator of (1) is the number of all nodes in a given mesh. The idea of (1) is presented on a simple example shown in Fig. 2. For this example $OB = (6 + 9 + 15)/42 = 0.714$

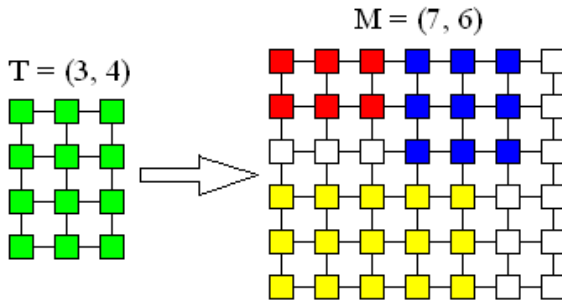


Fig. 2. Locked state – there is no possibility to allocate task T on mesh M

The main introduced index of performance of task allocation process is the objective function SO, called the average network load, defined by (2). It is the arithmetic mean (m is the number of occurrences of ‘the locked states’) of all measured OBs in a given allocation process expressed in percent.

$$SO = \frac{\sum_{i=1}^m OB_i}{m} 100\% \quad (2)$$

As the quality indicator of task allocation process, the end time allocation, denoted as TIME (3) can also be considered. The value t_{end} is the time-period between the starting time of the allocation process and the time, when the last task from the list T is allocated.

$$TIME = t_{end} \quad (3)$$

3 The Known Allocation Algorithms – Related Work

First Fit [5]. This algorithm is firstly creating a *busy array* with the list of all busy nodes. The binary value indicates the state of the mesh (1 means busy and 0 means free). Using *busy array* it can create the *coverage array* with list of free submeshes. The disadvantage of using this algorithm is not connected with any fragmentation. Due to the many searches made in every step, the time-duration of whole algorithm is increasing. Nevertheless, the First Fit solution is the most common in use because of the simple implementation.

Two-dimensional Buddy Strategy [6]. The idea consists in allocation the constructed blocks. Every block is divisible on four smaller blocks called *buddies*. To allocate one task, the algorithm is looking for the smallest free block which can be fulfilled by a given task. Free blocks are divided into *buddies* until the algorithm finds the smallest *buddy*, where the task could be allocated. The disadvantage of this algorithm is big internal fragmentation caused by rounding up the size of submeshes and limitation to the square meshes.

Frame Sliding [7]. This algorithm is scanning the mesh using a virtual task, which has the same size as the task which is going to be allocated on the mesh. The virtual task, as a frame, is moving. The algorithm is checking, if all nodes in a given location of frame are free. The stop criterion of this algorithm is fulfilled when the frame finds free space or the frame scans all nodes in the mesh. The main disadvantage of this solution is big external fragmentation caused in orderly movement of the frame.

Adaptive Scan [7]. This solution also as First Fit, uses frame. It is also creating a *coverage list* of all busy submeshes. Algorithm is checking if nodes scanned by frame consist into *coverage list*. If all nodes are not listed in *coverage list*, the allocation of task is possible. Such an allocation process has very long time-duration and is totally useless in cases where the main mesh is very big.

Stack Based Allocation [8]. This approach can make a possibility to change the orientation of task if it is necessary. There is a creation of *stack* with candidates list and also the *coverage list*. Candidates form the free spaces around the already allocated space, so left, right, upper and lower candidate can be distinct. All of them are putting onto the stack. The algorithm finishes the work after dividing all free space on mesh into candidates. There is only one step afterward consisting in checking whether any candidate could be a submesh for the next task from the list.

LeapFrog [9]. This algorithm uses a special way of describing the main mesh. It is a *R-Array*. Every element of this array describes the ‘situation’ around the singular node. If the element is positive, it means that around the node there is a sequence of free nodes. If the element is negative, there is a sequence of busy nodes. Absolute value of the element gives the information about number of nodes in the sequence. If the task has size $w' \times h'$, then *R-Array* is scanned and checked whether a given element has a value smaller than w' . If all values are greater, the process of allocation ends. In the opposite case, the algorithm is skipping to the next node. This procedure is quick for big meshes. The disadvantage is complicated implementation.

Quick Allocation [9]. This algorithm is based on the First Fit algorithm. To decrease the time of allocation process, the algorithm is creating the collection (segments) of busy nodes in neighborhood for every row of mesh. Every segment shows the candidates, where the task could be allocated. The bottleneck of this solution is the time of dividing the free space on mesh into segments. The computational complexity is increasing with size of the mesh.

Best Fit [6]. This algorithm, very similarly to the First Fit, firstly creates *busy array* and *coverage array*. The difference is that the Best Fit algorithm is looking for all free submeshes and after that is choosing the best one. The external fragmentation is very low, but the time-duration is increasing due to the necessity of full scans of the mesh.

4 The Created Algorithms

4.1 2Side LeapFrog Algorithm

The created algorithm is a modification of a Leap Frog solution. The main modification is that the starting point depends on the index of task. If the task has even index then the algorithm starts working on the upper left corner of the mesh. Otherwise, the process begins on the lower left corner.

The flowchart of the algorithm is presented in Fig. 3, where w' and h' define the size of a task, w and h are the size of mesh, i and j are the coordinates of a node and *flag* is a binary value informing about that the task has changed orientation [5].

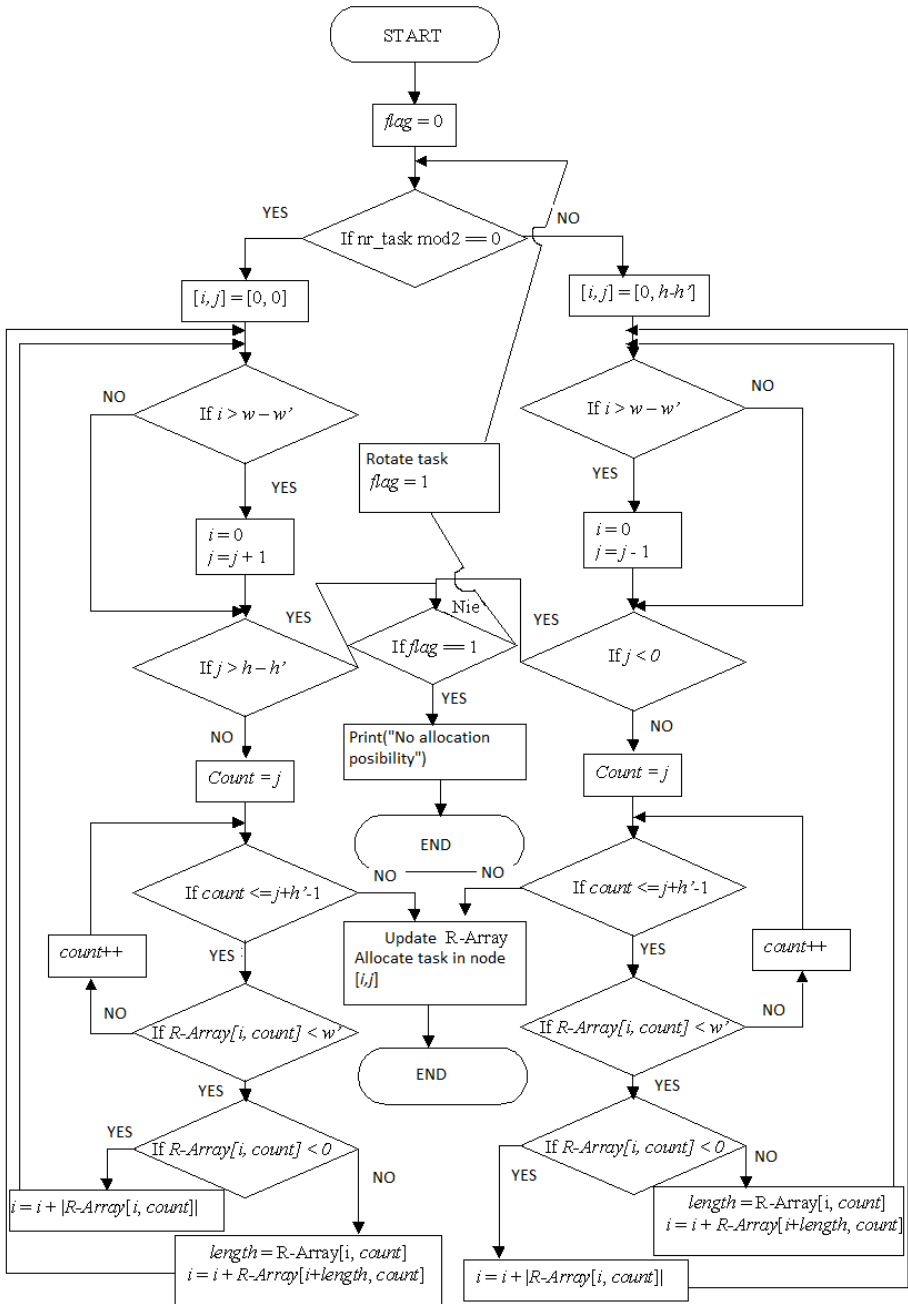


Fig. 3. Flowchart of 2Side Leapfrog algorithm

4.2 Q-Learning Based Algorithm

The idea of reinforcement learning [3] is used. The process of learning is based on studying the behavior of the system depending on the dynamic interaction with the environment. Interactions are taking place in time instants. The learner is observing every another state of environment. He behaves in accordance with the adopted strategy. After every action, the learner is receiving an award. The procedure of reinforcement learning is as follows. In every discrete time t :

- observe the current state x_t ;
- choose action a_t to make it in state x_t ;
- make action a_t ;
- observe the reinforcement r_t and another state x_{t+1} ;
- learn from experience (x_t, a_t, r_t, x_{t+1}) .

The Q-learning algorithm [10] consists in learning how to estimate the optimal value of action function (Q) by using strategy π , i.e., making an action $a_t = \pi(x_t)$. The value of action function Q is updated following the expression (4).

$$Q(x_t, a_t) = Q(x_t, a_t) + \beta[r_t + \gamma \max_{a'} Q(x_{t+1}, a') - Q(x_t, a_t)] \tag{4}$$

The factor β is a learning coefficient and its value is from range from 0 to 1. This is the rate of the size of modification. The discount factor γ is also a value from range from 0 to 1. It is a variable describing the quality of award.

We have proposed to adopt the task allocation problem in the following way:

- set X is a collection consisted of every possible permutations of tasks (the sequences in the queue);
- collection of every actions A consists of all possible changes between tasks on the task list;
- award r_t is the value of the objective function SO after action a_t in the state x_t ;
- action function Q is represented by a table for every state x and action a ;
- with some probability ϵ (greedy parameter) greater than 0, an action is chosen due to the uniform distribution and with $1-\epsilon$ probability if the action need to be greedy action following the formula (5):

$$\pi(x, a^*) = \begin{cases} \frac{1-\epsilon}{|\text{Arg max}_a Q(x,a)|} + \frac{\epsilon}{|A|}, & \text{where } a^* \in \text{Arg max}_a Q(x, a) \\ \frac{\epsilon}{|A|}, & \text{otherwise} \end{cases} \tag{5}$$

5 Investigation

5.1 Investigation Scenario

We tried to design as versatile simulation environment as possible, to be able to evaluate all combinations of parameters for various problem instances. As a result we

have developed a console application (to be run under Microsoft Windows OS), written in the C++ language, with various abilities to read parameters for the experiments and to write their results. We divided our investigation into two parts.

In the part one, we compared task allocation algorithms: First Fit, Frame Sliding, Adaptive Scan, Best Fit and LeapFrog BF described in Section 3 and the created algorithm 2Side LeapFrog described in Section 4. The three complex experiments for the three problem instances were carried out. The distinct categories of instances were: (i) tasks relatively small (as compared with the mesh size), (ii) tasks relatively large, and (iii) mixed tasks (mixed small and large). The sets of small category contained tasks with sizes (of their three parameters w , h , t) generated at random from the range 2-10. The sets of large category contained tasks with sizes (of their three parameters w , h , t) generated at random from the range 4-12. The sets of mixed category contained tasks with sizes (of their three parameters w , h , t) generated at random from the range 2-12.. Each complex experiment was repeated for 100 different sets of tasks generated at random (but the same for every allocation algorithm to be compared) and means values of indices of performance were calculated.

In the part two, the best algorithm from the first part was chosen as the auxiliary algorithm for Q-learning Based Algorithm. In this part, the influence of internal parameters of Q-learning Based Algorithm on its efficiency as well as the influence of category of sets of tasks was investigated. Moreover, the comparison of efficiencies of allocation processes performed for 2 Side LeapFrog without and with Q-learning Based Algorithm was made.

5.2 Part One. Comparison of the Allocation Algorithms

The average network load. We made investigation for different categories of sets of tasks. The obtained results are shown in Fig. 4 – Fig. 6 (where SO defined by (2) is denoted here as OS), respectively.

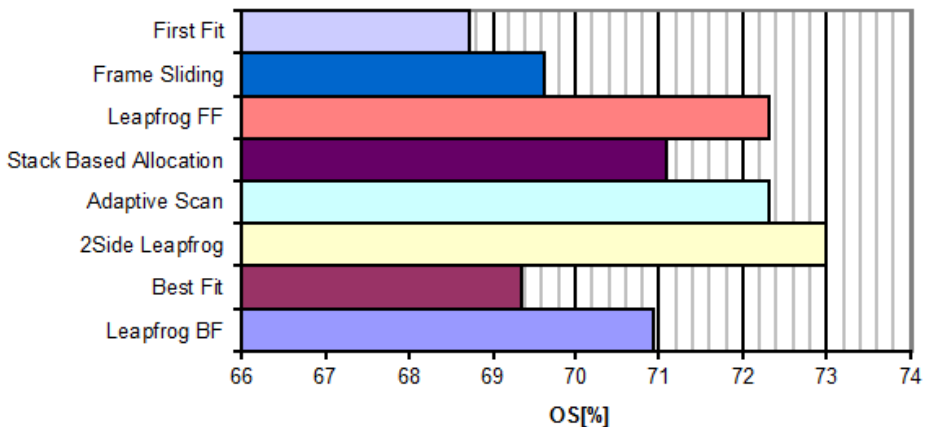


Fig. 4. Comparison of allocation algorithms efficiency for small tasks

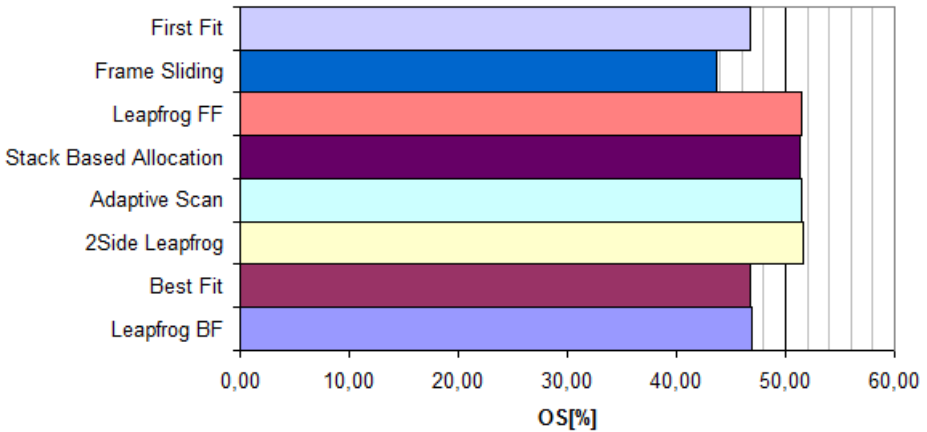


Fig. 5. Comparison of allocation algorithms efficiency for large tasks

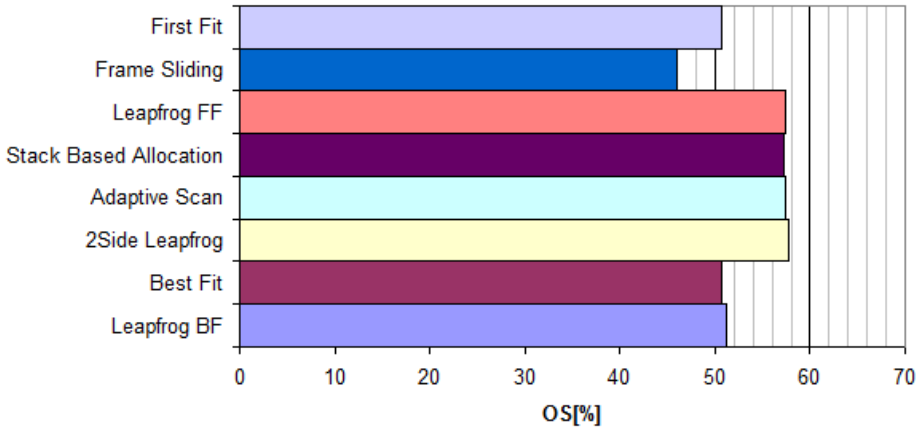


Fig. 6. Comparison of allocation algorithms efficiency for mixed tasks

It may be observed that for each category of sets of tasks the best solution was received for 2Side Leapfrog algorithm, especially this algorithm outperformed others for sets of small tasks. We were looking for the highest value of OS and the smallest value of t_{end} .

Time Duration. In this experiment we were looking for the shortest TIME defined by (3). The same sets of tasks were allocated on the meshes of different sizes (see Fig. 7). The best solution was received for Stack Based Allocation algorithm. However, the 2Side Leapfrog algorithm gave similar results for large range of mesh sizes.

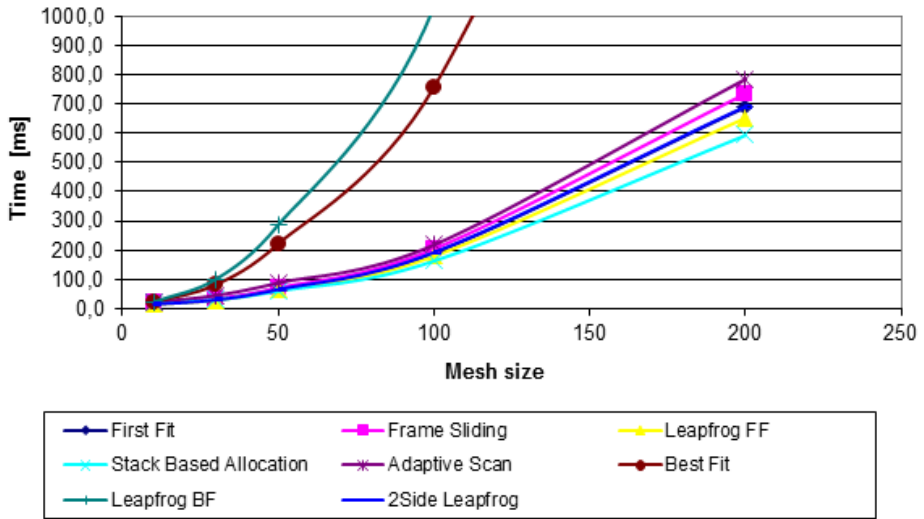


Fig. 7. Time duration of allocation process for different algorithms

5.3 Part Two. Evaluation of Q-learning Based Algorithm.

Adjusting Internal Parameters. Firstly, we were studying the relation between the number of iteration of Q-learning algorithm and the index of performance SO . We figure out that the duration time increases in proportion to the number of iteration (obvious observation) and that the efficiency of Q-learning algorithm is very random for small number of iterations. The situation became stable (see Fig. 8) above the number of 2000 iterations. Therefore, for further research we used the number of iterations equal to 3500, because greater number did not give higher efficiency but could cost more.

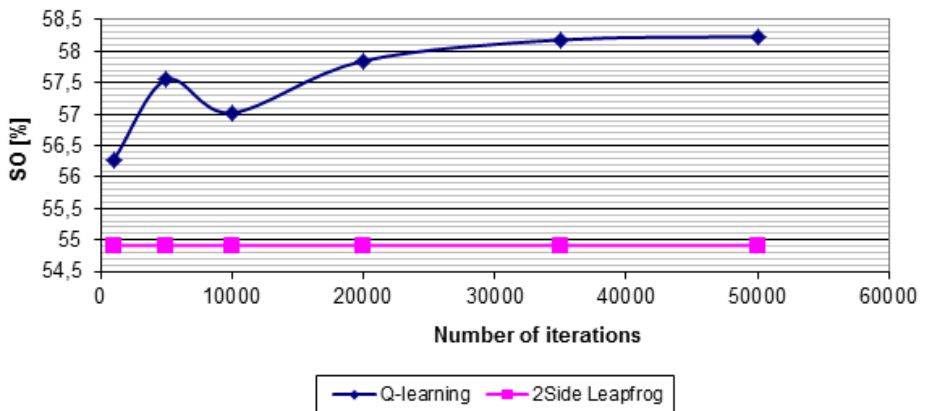


Fig. 8. The efficiency of Q-learning Based Algorithm in relation to the number of iterations

Next, we tested the influence of greedy ϵ - coefficient. Next, we checked the influence of the greedy coefficient on the efficiency of Q-learning algorithm. It may be observed (Fig. 9) that the algorithm works better with lower range of greedy parameter. The reason is that for greater greedy coefficients, the algorithm is oscillating between local maxima and it is very hard for the algorithm to change the location. With lower range of greedy coefficient the algorithm has bigger possibility to explore the whole collection of states.

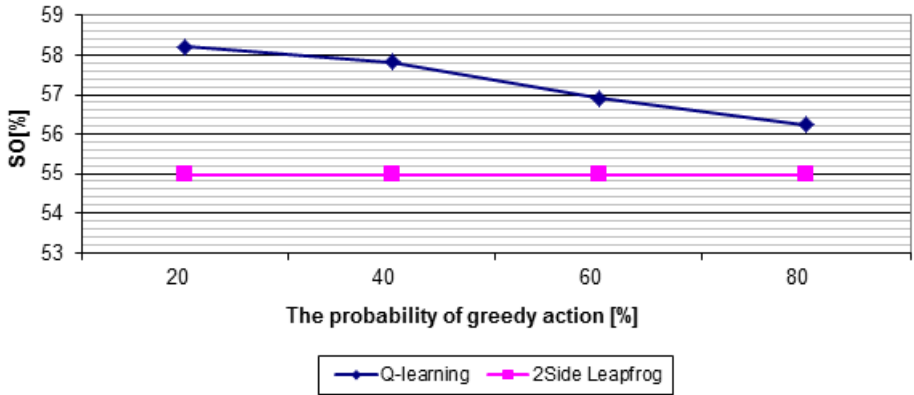


Fig. 9. Influence of greedy coefficient on the efficiency of the Q-learning Based Algorithm

Efficiency of the Q-learning Based Algorithm. The goal of this investigation was, from one side, to compare the behavior of Q-learning Based Algorithm in relation to the categories of sets of tasks, and from the other side, to check, whether the Q-learning Based Algorithm is better than only the auxiliary one. The results are shown in Fig. 10. It may be observed that for small and large tasks the improvement is less than 1%, but for mixed tasks is almost 4%.

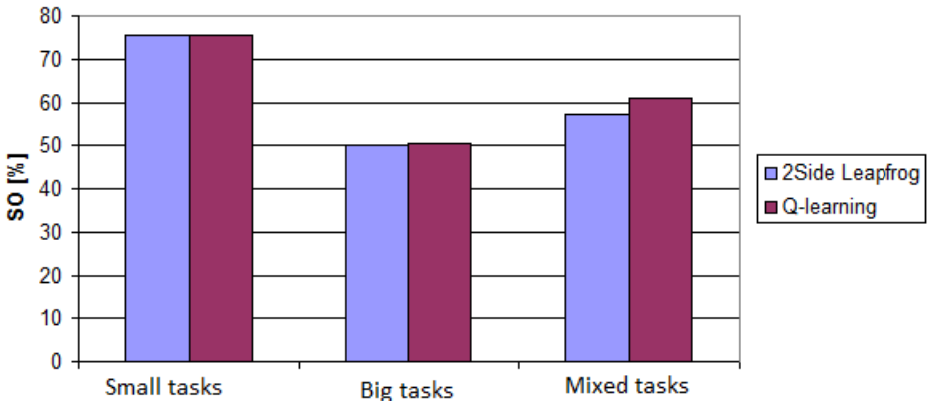


Fig. 10. Efficiency of Q-learning Based Algorithm for different categories of tasks

6 Conclusion

We have created two algorithms for solving the problem of task allocation on a multi-processor mesh, implemented an experimentation environment for testing them and performed research with it.

The experiments shown, that in general, the proposed 2 Side LeapFrog algorithm together with Q-learning Based Algorithm perform well in solving the considered task allocation problem. The advantage of using such approach is remarkable for sets of mixed tasks and not very large meshes. Moreover, taking into account the introduced measure of efficiency, it was observed that the 2 Side LeapFrog outperformed other known task allocation algorithms.

It definitely can not be said that the problem has been fully explored and researched. We are planning in further research: (i) to adapt more deeply ideas presented in [11] for performing future experiments, (ii) to construct a versatile testing environment, (iii) to use the full potential of Q-learning idea, and (iv) to implement new allocation algorithms based on the neural networks.

References

1. Ding, J., Bhuyan, L.N.: An Adaptive Submesh Allocation Strategy for Two-Dimensional Mesh Connected Systems. Texas A&M University (2002)
2. Das, C.R., Sharma, J., Pradhan, D.K.: A Fast and Efficient Strategy for Submesh Allocation in Mesh-Connected Parallel Computers. In: Proc. of 5th IEEE Symp. Parallel and Distributed Processing (1993)
3. Weber, C., Elshaw, M., Mayer, M.: Reinforcement Learning, Theory and Applications. I-Tech Education and Publishing, Vienna (2008)
4. Yoo, S.M., Youn, H.Y., Shirazi, B.: An Efficient Task Allocation Scheme for 2D Mesh Architectures. *IEEE Transactions on Parallel and Distributed Systems* 8 (1997)
5. Goh, L.K., Veeravalli, B.: Design and Performance Evaluation of Combined First-Fit Task Allocation and Migration Strategies in Mesh Multiprocessor Systems. *Parallel Computing* 34, 508–520 (2008)
6. Chuang, P.J., Tzeng, N.F.: An Efficient Submesh Allocation Strategy for Mesh Computer Systems. In: Proceedings of 11th Int. Conf. on Distr. Comp. Systems (1991)
7. Byung, S., Yoo, B.S., Das, C.R.: A Fast and Efficient Processor Allocation Scheme for Mesh Connected Multi-computers. *IEEE Trans. on Computers*, 46–59 (2002)
8. Koszalka, L.: Simulation-Based Evaluation of Distributed Mesh Allocation Algorithms. In: Thulasiraman, P., He, X., Xu, T.L., Denko, M.K., Thulasiram, R.K., Yang, L.T. (eds.) *ISPA Workshops 2007*. LNCS, vol. 4743, pp. 335–344. Springer, Heidelberg (2007)
9. Wu, F., Hsu, C., Chou, L.: Processor Allocation In the Mesh Multiprocessors Using the Leapfrog Method. *IEEE Transactions on Parallel and Distributed Systems* 14 (2003)
10. Watkins, C.J.C.H., Dayan, P.: Q-learning. *Machine Learning* 8, 279–292 (1993)
11. Koszalka, L., Lisowski, D., Pozniak-Koszalka, I.: Comparison of Allocation Algorithms for Mesh Structured Networks with Using Multistage Simulation. In: Gavrilova, M.L., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) *ICCSA 2006, Part V*. LNCS, vol. 3984, pp. 58–67. Springer, Heidelberg (2006)

Follow-U: A Distributed Ubiquitous Healthcare System Simulated by MannaSim

Maria Luísa Amarante Ghizoni, Aduino Santos, and Linnyer Beatrys Ruiz

Manna Research Group of Invisible Computing Engineering
State University of Maringá – Brasil
{malu.ghi, adauto.info2006, linnyer}@gmail.com

Abstract. The monitoring of assisted living requires the existence of tools that capture vital signs from the user and also as well as environment data. Ubiquitous healthcare aims to use technologies such as: wireless sensor networks, context awareness, low-power electronics, and so on. This work employs Information and Communication Technologies (ICT) to provide a technologically and socially acceptable solution to aid elderly and disabled people to live a more independent life as long as possible, and to maintain their regular everyday activities at their own home. As a study case, we have implemented a simulation of a smart home, called Follow-U, comprised of networked computational devices. Follow-U can be used in different ambients such as hospitals, houses, rest homes, gyms, so on. In particular, we present the simulation in a house plan.

Keywords: Ubiquitous Healthcare, simulation, elderly healthcare.

1 Introduction

Ubiquitous Systems are equipped with user-friendly interfaces applicable in different contexts, which are present in the everyday life of common citizens. Wireless Sensor Networks (WSNs) are covered by the concept of Ubiquitous Computing. They are basically a network made up of many sensor nodes with low computational power, working collaboratively to accomplish sensing, processing and communication of data for a particular environment or phenomenon. [5]

Ubiquitous computing is a technological concept that encompasses Mobile Computing and Pervasive Computing, with the goal of making computers ubiquitous and invisible to people. Thus, human-computer interaction is made through natural behaviors and actions, such as the use of gestures, body movements, speech, and by external variables captured from the environment. In this context, the use of a new device must not require a great deal of learning experience.

Currently, there are several projects that carry out the simulation of WSNs, such as the ns-2 Network Simulator [14] Simulating TinyOS Networks (TOSSIM) [13] Java in Simulation Time (JIST) [10], Algorithms for Network Simulator (SinalGo) [11], among others. The use of these simulators provides a wide range of possibilities.

Through a simulated environment, it is to add new components to the network without requiring the physical components themselves. Also, it is possible to test different distances, protocols, forms of sensing and dissemination of information and even the manipulation of data obtained from the environment. In this work we chose the Network Simulator, along with the MannaSim package [12], which adds modules for the simulation of WSNs.

Ubiquitous systems must provide their services all the time and everywhere. In order to increase transparency and personalization, ubiquitous applications are normally context-aware, i.e., they use information about entities of interest to adapt their services. Since they are connected to everyday elements, such systems are frequently shared by two or more users, who may provide conflicting contextual data. Therefore, these systems can reach an inconsistent state, in which they are unable to decide how to perform their intended adaptations [8].

This work proposes a service-based Ubiquitous Healthcare solution in order to provide support for independent living using different technologies to gather, process, generate, and disseminate data about humans, the environment, artifacts, and intangible parameters. In general, these systems communicate through wireless technologies, in cooperative or competitive stance using collective context information.

Many theoretical works have been proposed using Wireless Sensor Networks (WSN), context awareness and other technologies in the design of solutions for ubiquitous healthcare. These work use simulations in the evaluation of their proposals.

Follow-Us is deployed in a home equipped with information and communications technologies in order to provide quality of life for the elderly. Using Follow-Us, healthcare personnel are not required to be physically close to the patients, ensuring independent living and safety.

The remainder of this article is organized as follows. We define the simulation chosen as a case study. Then, we present one of the important contributions of this work, the definition of U-healthcare services developed based on autonomic computing, wireless sensor networks, domotics and context-awareness concepts. Then, we present the results of Simulation. Finally, we present our concluding remarks.

2 Application Scenario

Oftentimes, family can be working or is simply too far to assist a loved one with those daily seemingly easy tasks that may have become too difficult, either physically or mentally. Follow-Us can prevent social isolation and support by maintaining a multi-functional network around the individual to promote a better and healthier lifestyle for individuals at risk or that need special care.

The proposed application represents a class of U-healthcare applications with enormous potential benefits to the whole scientific community and society. Instrumenting spaces with networked sensor nodes, and low-power devices can promote a shared smart home that helps the family, gives freedom to a loved one and maintains welfare without burdening the health system. Besides this, the proposed solution can contribute to reduce hospitalization or specialized institutions cost.

As a proof of concept for our application, we simulated this scenario using the Network Simulator (ns-2) using MannaSim. This simulation is described in the next Section.

2.1 Network Simulator (ns-2)

The ns-2 [14] is an event simulator for testing and researching networks, currently maintained by UC Berkeley, used primarily in academia. It provides support for simulating transmission and routing protocols for wired and wireless networks, schedulers and their policies in line, traffic characteristics and much more. Ns-2 uses two languages, C++ to create and manipulate the structures used in the network (agents, protocols, transmission methods, byte manipulation, among others) and the OTcl (Object-oriented Tool Command Language) scripting language, which is used for setting up simulations.

By design, ns-2 is efficient, counting on the performance benefit of C++ for handling low level aspects while also easing the development process, by using OTcl, which is an interpreted language, and therefore there is no need for a new build if any simulation parameter changes. The execution of the script through the simulator produces an output file known as a trace file. In this file a lot of information about the work of the network may be gathered, such as: energy level, the protocol behavior, description of events, position of components, due to the fall, the path of the package and other information's.

To develop this work, we used ns-2 (version 2.29) with the MannaSim package. The following packages must also be installed: Tcl (Tool Command Language), Tk (Toolkit graphical user interface), OTcl (Object-oriented Tool Command Language) and Tclcl (Classes with Tool Command Language).

2.2 MannaSim

MannaSim [12] is a simulator for wireless sensor networks based on the Network Simulator, which extends and introduces modules for this particular type of network. Moreover, it has a front-end tool for TCL code generation (MannaSim Script Generator), providing ease and accuracy in the development of scenarios and in particular providing a comprehensive set of algorithms and protocols.

First the main settings are chosen in the simulation, such as the transport protocol (TCP and UDP), routing protocol (AODV, DSR, TORA, LEACH, Directed Diffusion and DSDV), the physical layer (914MHz Mica2 and Lucent WaveLAN DSS), the propagation of radio the signal, the buffer size, the size of the area, the simulation time and other settings.

After the choice of key parameters, it is necessary to decide the features present in the common nodes, access points and cluster heads, as the number of devices on the network reach, level of initial energy, transmission range, time and form of dissemination and sensing, and data generators (e.g. temperature and carbon dioxide).

3 Follow-Ups: Putting It All Together

Follow-Ups is a distributed ubiquitous healthcare system that is intended to be used by elderly and disabled people and aims to assist them to live a more independent life as long as possible and to maintain their regular everyday activities at their own home.

Like in [1], our work is about a house that aims to provide self-management style of health and safety for its inhabitants, for example, self-health monitoring and self-medication measurements. The house is connected in an intelligent way to access the

Internet to send messages to the back-end professionals, which are continuously informed about the status of the monitored residents to deal with emergency situations that require intervention.

Work described in [4] takes advantage of advances in electronics technology and low-power sensors, which lead to small-sized medical body sensors and actuators that are capable of collecting physiological data from the body of the local monitored and possibly deliver some drugs. This is done through the deployment of a wireless body area network (WBAN). What is seen as a good step to be incorporated into our future work.

In particular, we decided to monitor the motions in the elderly population because many common complaints, such as dizziness and loss of balance. Despite the tremendous advances in low-power and nano-sized electronics, invisible sensor nodes are not yet commercially available. The Follow-Ups came to provide a way to integrate many kinds of artifacts, environments and also the elderly people with sensors that are among the objects and people, e.g., in their clothes.

We have used the MicaZ sensor node platform as a body wireless sensor node and the Sensor board (MTS300). This sensor board is composed by accelerometers and microphones that are fixed on the body (shoulder, chest and legs). In [9], we proposed a tool that uses a WSN to perform patients' movement detection, assisting the study of abnormal cardiac rhythm.

The Follow-Ups prototype implements a wireless sensor network using MicaZ sensor node platform and MTS 300 sensor board in order to monitor the ambient considering thermal comfort, lighting, and humidity. This way, the smart home can decide to perform some interpretation and actuations services, e.g., to open or close the windows in order to promote air renewal, to control of temperature without turning on air conditioning.

The prototype also decides to water plants according to data gathering about soil humidity of the plants. In this case, the prototype uses the WaspMote Sensor Node with soil moisture, leaf wetness sensor device.

Considering the audiovisual perception service, if any intervention is needed, when abnormal conditions are detected (e.g. elderly fall), the body sensor nodes also capture the voice or breath, and a visual sensor node is triggered to make an image and send all data about the loved one's accident to family or rescue staff.

The Follow-Ups can use wireless sensor networks protocols to route the data into the WSN. In particular we have performed directed diffusion, LEACH, zigbee and 802.11. Outside of the WSN, the Follow-Ups can use SMS or social networks tools, such as Orkut, Facebook, or Twitter in order to send data to relatives. The family or the health professional may receive, with the desired periodicity, a report, which has processed intelligently the information gathered from sensors about the elderly person's activity and behavior. This data can be used for trend analysis and medical research.

We have adopted the idea to open and close doors and windows with a servomotor stuck on the house wall pulling each tab of the window by a small steel cable connected to a rod fixed on the window. To receive the command to open and close the window, a servomotor was connected to an Arduino, which has an ATMEL microcontroller 186 and a door that connects the PWM servo. This microcontroller is connected to a ZigBee module, which performs communication with the ZigBee network. Just as the doors and the windows, lamps operate similarly using Arduino and ZigBee, but they were connected to relays in order to use an AC 110V. In this prototype version, four windows, one door and two lamps are managed in autonomic way.

A web service system permits family members from a computer or device that has Internet access, such as a cell phone or a tablet, to ask for information, images, so on. Also, the web service also permits to control operation of electrical equipments, such as electric iron, air conditioning, fans, televisions, and showers, for assisting the elderly in daily tasks. Each device is a button that can be clicked in order to change its state, as long as the relative has the required permission. In this homepage, graphs are presented showing the history of temperature and humidity of the house, as well as, elderly information.

In this way, it is possible the occurrence of conflicts because the limitations due to ageing ask an action and on the other side the need of security of older people ask an opposite action. In [8] we proposed a novel methodology called CReMe (Conflict Resolution Methodology) that detects and solves conflicts of interest for ubiquitous context-aware applications with different characteristics. Besides, the developed approach considers the trade-off between users' satisfaction and resources consumption in order to select and apply a conflict resolution algorithm. Results obtained showed that the proposed solution is flexible, dynamic, and able to provide users' satisfaction as well as to save system resources.

Considering the abstraction service, the elderly receives some support from Follow-Ups in order to help in his daily life, e.g., the system manages electrical appliances on/off, main door opened. The solution also has a friendly tool to annotate the elder's appointments and can advise their fulfillments (e.g, take medicine, watering plants, exercise of physiotherapy in the rehabilitation of postural control, associated with senescence). In this case, messages are shown in TV, tablets, mobile phones, and special sounds.

Internal contexts, like feelings and physiological needs, could also be relevant to adaptation decisions. Nevertheless, these contexts are hard to be obtained because, usually, they can be not gathered by physical sensors. The preliminary qualitative results show that the inference system is accurate when the sentences are well-formed and do not present types. The smart home can post tweets to the supermarket, the pharmacy, the family, so on. It can also decide whether to turn on the lights providing an opportunity to build intelligent and calm interfaces. The shared smart home solution monitors everyday items in order to warn the patients, friends, family or medical service about the need for intervention when abnormal conditions are detected as well as to collect data for trend analysis and medical research. It adopts the incorporation of ambient sensors, context awareness for improved sensing and episode detection.

The Follow-Ups can be customizable for any physical environment (houses, hospitals, elderly houses, classes, offices, and so on) and has been designed based on natural interfaces, hands-free equipment's suitable for elderly with decreased hand-eye coordination and vision, and devices capable of performing useful computing to the people around.

4 Simulation

As in [2] we use some criteria to evaluate our self-stimulation, the errors and energy spent. For evaluation, we simulate a house plan, with just one inhabitant being monitored in his everyday life, as in Figure 1 which shows the division of 29 common nodes and three access points in the monitored area.

Differently from [6], that the information is stored in XML archives, our simulation returns a Tcl archive that is stored in a mysql database. However, both smart-house simulations, ours and theirs, were developed based in an event and message structure.

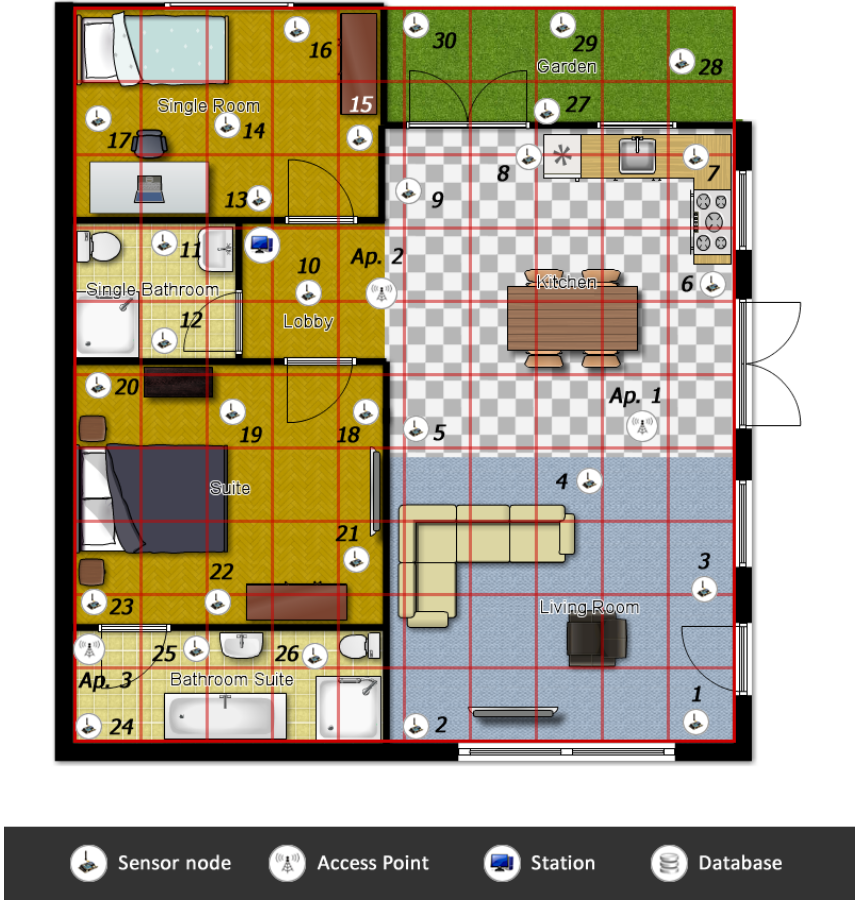


Fig. 1. Division of common nodes and access points in the environment

4.1 Energy Level

In Common Nodes

During operation of the 33 simulations, it was found that the level of remaining energy of sensor nodes ranged from 24.35657 to 24.43624 joules and joules average was 24.40743. Thus, the average consumption of network operation in its lifetime, was 5.596 joules.

Then the graph shown in Figure 2 illustrates the level of energy in the common nodes.

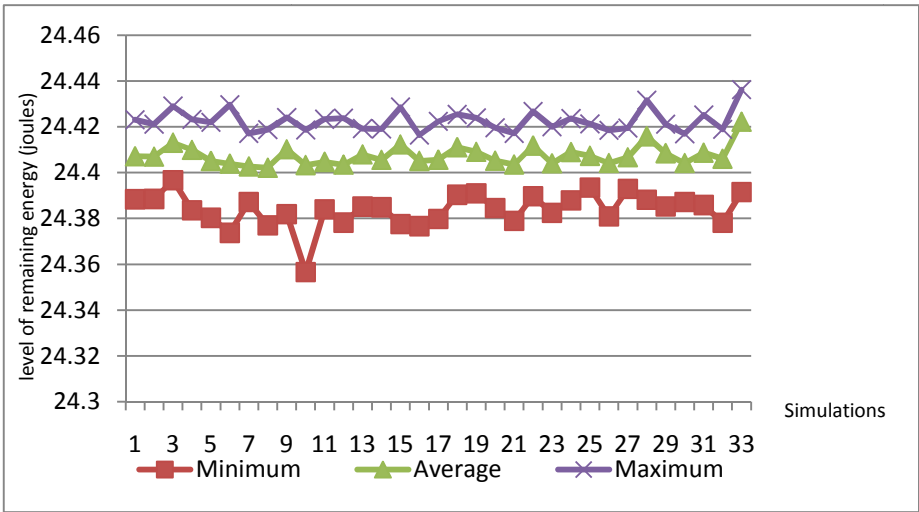


Fig. 2. Level of Energy Common Nodes

Access Points

During operation of the 33 simulations, it was found that the average remaining energy of the access points ranged from 34.883 to 91.931 joules and the average stood at 60.7848 joules. Thus, the average consumption of network operation in its lifetime, was 39.2152 joules.

Then the graph shown in Figure 3 illustrates the energy level of the access points.

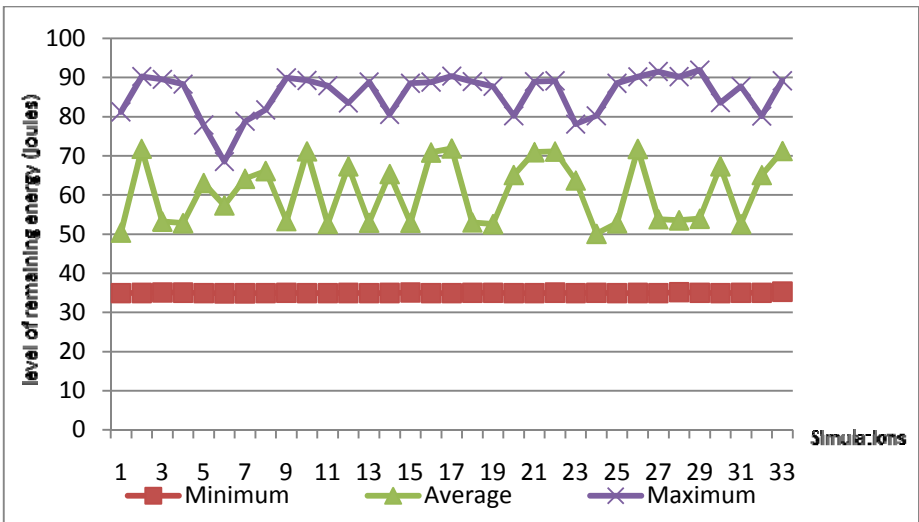


Fig. 3. Remaining energy level of the access points

4.2 Number of Errors

During operation of the 33 simulations, it was found that the amount of errors in a simulation, ranged from 445,053 to 455,977 and the average error was in 448,883.3 errors. Next, Figure 4 illustrates the variation of the number of errors among simulations and Figure 5 illustrates the division of errors among access points and common nodes.

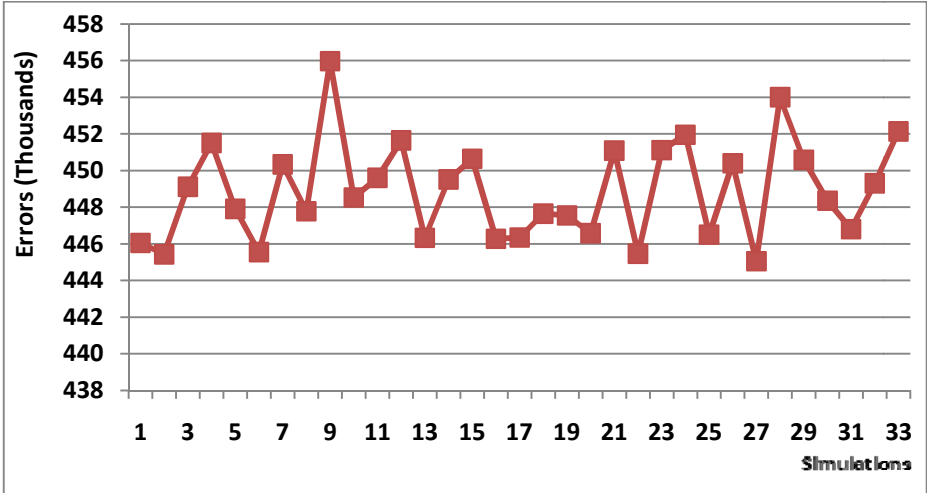


Fig. 4. Number of errors among simulations

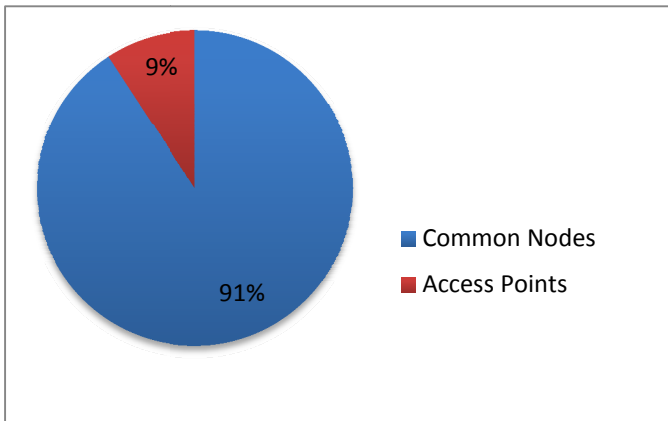


Fig. 5. Division of errors among the access points and common nodes

Common Nodes

During operation of the 33 simulations, it was found that the amount of errors in a simulation ranged from 398,783 to 416,857 and the average error was in 407,556.606 errors. Next, Figure 6 illustrates the number of errors in common nodes.

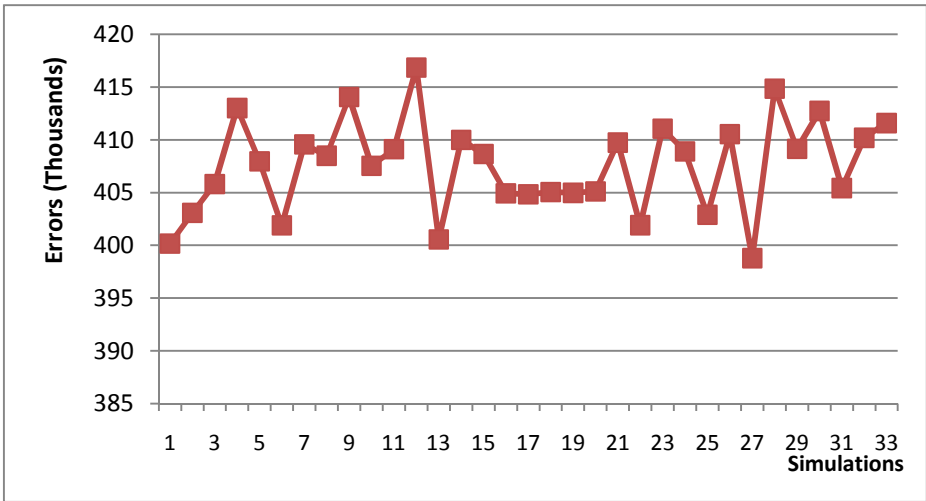


Fig. 6. Number of errors in common nodes

Access Point

During operation of the 33 simulations, it was found that the amount of errors in a simulation ranged from 34,791 to 46,270 and the average error was in 41,326 errors. Next, Figure 7 illustrates number of error in access point.

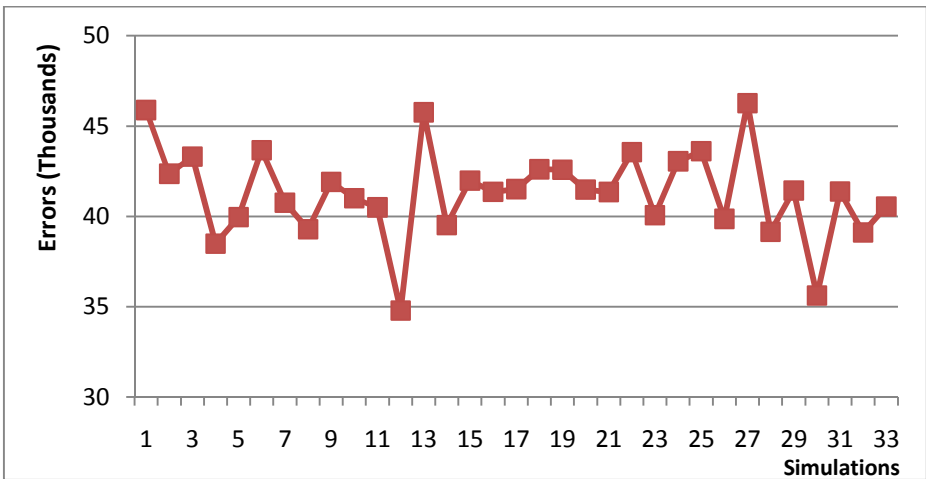


Fig. 7. Number of errors in access points

4.3 Types of Errors

Access Point

In the experiments, we found that we shared 98% of the errors (1,341,910) are collision packet errors, such errors are not reported in the trace file representing 2% of the errors and the others combined do not reach 1%.

Table 1 shows the division of the types of errors found in access points.

Table 1. Types of errors found in access points

Typesoferrors	NumbersofErrors
Uninformed	21,210
ARP	206
CBK	146
COL	1,341,910
NRT	10
RET	298
Total:	1,363,780

Common Nodes

In the experiments, we found that we shared 94% of the errors (12,619,826) have listed their cause in the simulation trace file, the remaining errors are mostly mistakes of packet collision (COL - 5%) and errors caused by excess packets in the queue in-interface (IFQ - 1%).

However, removing errors due to non-informed, it is observed that 73% are errors related packet collision (COL), 23% are IFQ errors, 2% errors are due to excessive attempts were sending (RET), 1% error callback (CBK), 1% are errors caused when there is an excess of ARP packets in the queue (ARP) and the absence of routing errors is less than 1%.

Below, Table 2 and Figure 8 show the breakdown of types of errors we found in common nodes.

Table 2. Types of errors found in Common Nodes

Typesoferrors	NumbersofErrors
Uninformed	12,619,826
ARP	2,773
CBK	9,800
COL	603,044
IFQ	193,115
NRT	1,810
RET	19,000
Total:	13,449,368

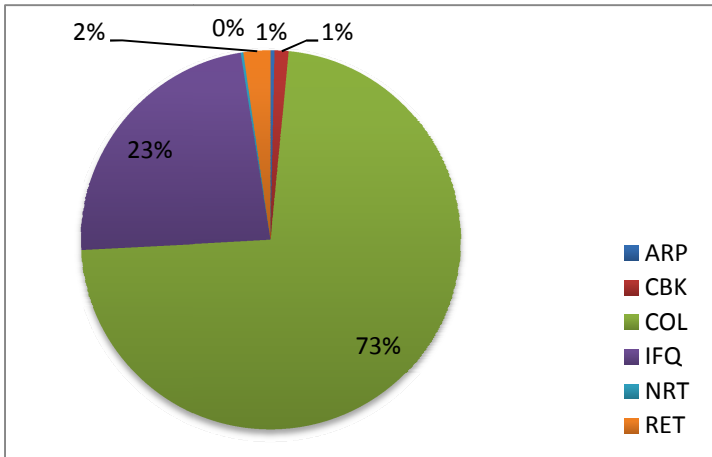


Fig. 8. Division of error types in common nodes (except for non-reported errors)

4.4 Division of the Evaluations Events

During the operation of 33 simulations were verified 82,430,928 events that are divided on the transmission of information (49%), receiving (33%), when an error occurs (18%) and routing of messages. The following Table 3 and Figure 9 show the division of simulations' events.

Table 3. Division of the simulations' events

TypeofEvent	Average	Total
Error	448,883.273	14,813,148
Routing	13,596.636	448,689
Receipt	817,909.303	26,991,007
Dispatch	1,217,517.697	40,178,084
Total:		82,430,928

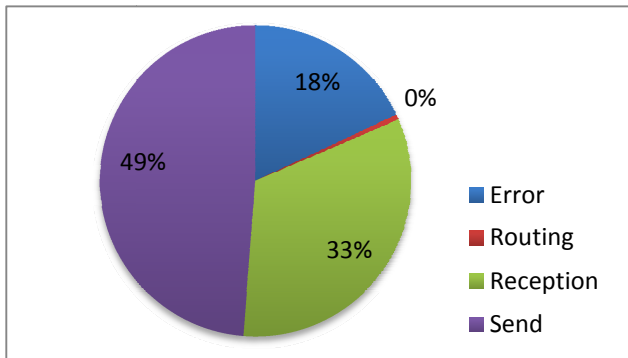


Fig. 9. Division of the simulations' events

4.5 Operation of the Network in Its Lifetime

Analyzing the behavior of the network during its lifetime, it was found that the first network has a second peak in the number of messages (33,912 posts), but in a few seconds the number of messages falls dramatically, and in second four it is already in 16,384 messages.

This trend tends to remain stable up to 149 seconds, which marks the end of the run of the simulation. Below, Figure 10 illustrates the operation of the network during its lifetime.

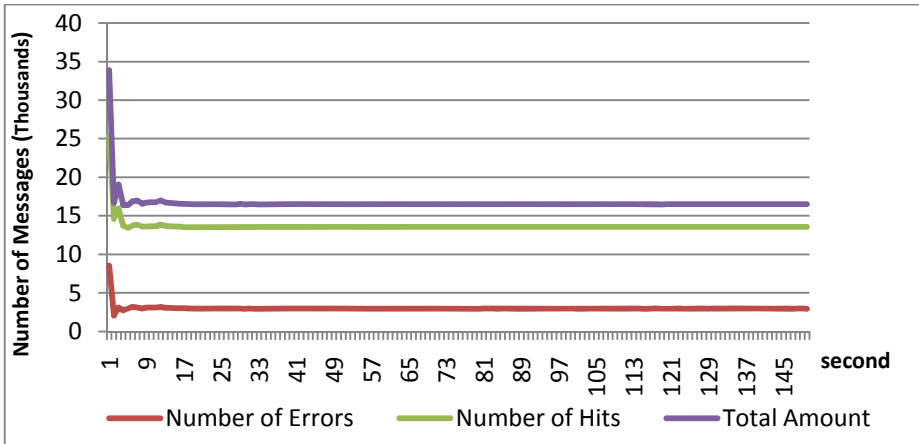


Fig. 10. Operation of the network in its lifetime

5 Conclusions

With the various simulations we were able to see the convergence of the errors so we can improve our WSN, increasing the number of access points and transmitting only the necessary number of messages in order to decrease the large number of errors, before deploying it in a real environment. The consequences of the large number of errors could be devastating in a real environment because an unreported message could cause suffering to the elderly. We note that the most common mistakes happen in common node, which most of these errors are related to packet collision. We also noticed that they occur in sending messages in most cases.

We also know that energy is a critical resource in WSNs. The rate of energy consumption is referred to as the power. If the power is not managed efficiently, the lifetime of the power supply, such as batteries, will be shortened and the longevity of the network will suffer. So is possible to see how much energy would be spent in each node and access point by simulations. We have worked with them in other opportunities as well [3][7].

As a case study, in this work we proposed a shared smart home simulation, named Follow-Ups, that coordinates elderly care on a full time basis, helping the family with

insider information and the possible adaptation according to the elder's and other resident's needs. Social, ethical, security and privacy issues related to continuous monitoring of people, storing and analyzing the data and how to verify the safety, security and privacy aspects of the system are future work.

This has been a particular experience where we have used different technologies to experiment the advantages and benefits of applying WSNs, collective context-awareness and other methodologies to develop and deploy a real U-healthcare solution.

Acknowledgments. Our thanks to CNPq, Capes and INCT Namitec for helping us financially.

References

1. Agoulmine, N., Deen, M.J., Lee, J.-S., Meyyappan, M.: U-Health Smart Home. *IEEE Nanotechnology Magazine* 5(3), 6–11 (2011), doi:10.1109/MNANO.2011.941951
2. Cherkaoui, E.H., Agoulmine, N., Nguyen, T., Toni, L., Fontaine, J.: Taking Advantage of The Diversity in Wireless Access Networks: On the Simulation of a User Centric Approach. In: 2011 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 1021–1028 (2011), doi:10.1109/INM.2011.5990516
3. Foleiss, J., Faustino, A., Ruiz, L.B.: An Experimental Evaluation of Compiler Optimizations on Code Size. In: Proceedings of the Simpósio Brasileiro de Linguagens de Programação, São Paulo (2011)
4. Kim, J., Choi, H.-S., Wang, H., Agoulmine, N., Deerv, M.J., Hong, J.W.-K.: POSTECH's U-Health Smart Home for elderly monitoring and support. In: 2010 IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM), pp. 1–6 (2010), doi:10.1109/WOWMOM.2010.5534977
5. Li, Y., Thai, M.T., Wu, W.: *Wireless Sensor Networks and Applications*. Series: Signals and Communication Technology. Springer (2008) e-ISBN 978-0-387-49592-7
6. Park, J., Moon, M., Hwang, S., Yeom, K.: Development of Simulation System for Validating Contextual Rule in Smart Home. This paper appears in: The 9th International Conference on Advanced Communication Technology, vol. 2, pp. 1143–1146 (2007) ISSN: 1738-9445
7. Ruiz, L.B., Braga, T.R.d.M., Silva, F.A., de Assunção, H.P., Nogueira, J.M.S., Loureiro, A.A.F.: On the De-sign of a Self_managed Wireless Sensor Network. *IEEE Communications Magazine*, Estados Unidos 47(8), 1–10 (2005)
8. Silva, T.R.d.M.B., Ruiz, L.B., Loureiro, A.A.: Towards a Conflict Resolution Approach for Collective Ubiquitous Context-aware Systems. In: 12th International Conference on Information Integration and Web-based Applications & Services (iiWAS 2010), 2th International Conference on Information Integration and Web-based Applications & Services (iiWAS2010), Paris, ACM e Austrian Computer Society, New York-Vienna (2010)
9. Zorkot, A.C., Ruiz, L.B., de Assunção, H.P.: A Tool for Moviment Detection Using Wireless Sensor Network. In: XXXIII Integrated Simposium of Software and Hardware, 2006, Campo Grande. Proceedings of the Congresso Da Sociedade Brasileira de Computação 2006 (2006)
10. Cornell University, JIST - Java in Simulation Time, <http://jist.ece.cornell.edu/> (accessed June 19, 2011)

11. Distributed Computing Group, Sinalgo - Algorithms for Network Simulator,
<http://www.disco.ethz.ch/projects/sinalgo/> (accessed July 17, 2011)
12. Manna Research Group and Project Sensornet, MannaSim Framework,
<http://www.mannasim.dcc.ufmg.br/> (accessed August 15, 2011)
13. UC Berkeley Computer Science Division, Simulating TinyOS Networks,
<http://www.cs.berkeley.edu/~pal/research/tossim.html>
(accessed June 18, 2011)
14. University of Southern California, Network Simulator (ns-2),
<http://isi.edu/nsName/ns/> (accessed July 22, 2011)

Adaptive Dynamic Frequency Scaling for Thermal-Aware 3D Multi-core Processors

Hong Jun Choi¹, Young Jin Park¹, Hsien-Hsin Lee², and Cheol Hong Kim^{1,*}

¹ School of Electronics and Computer Engineering,
Chonnam National University, Gwangju, Korea

² School of Electrical and Computer Engineering, Georgia Institute of Technology,
Atlanta, Georgia, U.S.A.

chj6083@gmail.com, blueboy24@nate.com,
leehs@gatech.edu, chkim22@jnu.ac.kr

Abstract. 3D integration technology can provide significant benefits of reduced interconnection delay and low power consumption in designing multi-core processors. However, the 3D integration technology magnifies the thermal challenges in multi-core processors due to high power density caused by stacking multiple layers vertically. For this reason, the 3D multi-core architecture cannot be practical without proper solutions to the thermal problems such as Dynamic Frequency Scaling(DFS). This paper investigates how the DFS handles the thermal problems in 3D multi-core processors from the perspective of the function-unit level. We also propose an adaptive DFS technique to mitigate the thermal problems in 3D multi-core processors by assigning different DFS levels to each core based on the corresponding cooling efficiency. Experimental results show that the proposed adaptive DFS technique reduces the peak temperature of 3D multi-core processors by up to 10.35°C compared to the conventional DFS technique, leading to the improved reliability.

Keywords: Processor architecture, 3D integration technology, Thermal management, Dynamic frequency scaling, Multi-core processor.

1 Introduction

As process technology scales down and integration densities continue to increase, interconnection delay has become one of the major constraints in improving the performance of microprocessors[1-2]. As one of the most promising solutions to reduce the interconnection delay of multi-core processors, three dimensional(3D) integration technology has drawn significant attentions[3]. In 3D multi-core processors, multiple cores are stacked vertically in the same package and components on different layers are connected through direct through-silicon vias(TSVs)[4]. Therefore, the 3D integration technology reduces the interconnection delay of multi-core processors by

* Corresponding author.

decreasing the global wire length significantly compared to the 2D technology[5-7]. Reduced wire length also leads to the decreased power consumption [8-9]. For this reason, the 3D integration technology has drawn considerable attentions in designing recent multi-core processors.

Despite the benefits mentioned above, the 3D integration technology cannot be practical without proper solutions to the thermal problems in the processors. As mentioned in [10], the thermal problems are exacerbated in the 3D multi-core processors compared to the 2D multi-core processors mainly for two reasons. One is that the vertically stacked silicon layers cause rapid increase of power sources. The other is that the thermal conductivity of the thermal interface material(TIM) is lower than that of silicon and metal, resulting in a higher power density as more stacked TIM layers. Unfortunately, high temperature in the processor causes increased cooling costs, negative impact on the reliability and performance degradation. Therefore, more advanced cooling methods are required in designing upcoming 3D multi-core processors[2].

Dynamic thermal management(DTM) techniques using dynamic frequency scaling(DFS), dynamic voltage scaling(DVS), clock gating or computation migration have been proposed to relieve the thermal stress in 2D chips[11]. These techniques solve the thermal problems to a great extent by lowering the average temperature or by keeping the temperature under a given threshold[12]. However, conventional DTM techniques might not be sufficient to mitigate the thermal problems of 3D multi-core processors, because the peak temperature of 3D chips is much higher than that of 2D chips. Therefore, more aggressive DTM techniques for 3D chips should be investigated[13]. In our previous research, we analyzed the detailed thermal behavior of 3D multi-core processors where the conventional DFS technique is applied[2]. Based on the results in [2], in this paper, we propose an adaptive DFS technique to reduce the peak temperature of 3D multi-core processors. In the processor with the proposed adaptive DFS technique, the cores run with different DFS levels based on the corresponding cooling efficiency of the core, resulting in the reduced peak temperature.

The rest of this paper is organized as follows. Section 2 provides an overview of 3D integration technology and dynamic thermal management techniques. In Section 3, the proposed adaptive DFS technique is described. Section 4 provides the simulation methodology and the detailed results. Finally, Section 5 discusses our conclusions and future work.

2 Related Work

2.1 3D Integration Technology

3D die stacking is a new technology that increases transistor density by integrating two or more dies vertically[15]. Figure 1 shows the structural comparison between a 2D two-core processor and a 3D two-core processor. The 3D die stacking enables a significant reduction of wire length both within a die and across dies in the microprocessor. In the 3D microprocessor, blocks can be placed vertically on multiple dies to

reduce the wire distance, latency and power. For this reason, the 3D integration technology has the potential to change the processor-design constraints by providing substantial power and performance benefits compared to the 2D design technology. Despite the promising advantages, the 3D integration technology brings us some considerable issues. Thermal problem is one of the critical issues for the 3D microprocessor. The vertically stacked silicon dies causes rapid increase of power density, and the increased power density causes increased cooling costs, negative impact on the reliability and performance degradation. For this reason, more effective cooling techniques are required in designing upcoming 3D multi-core processors.

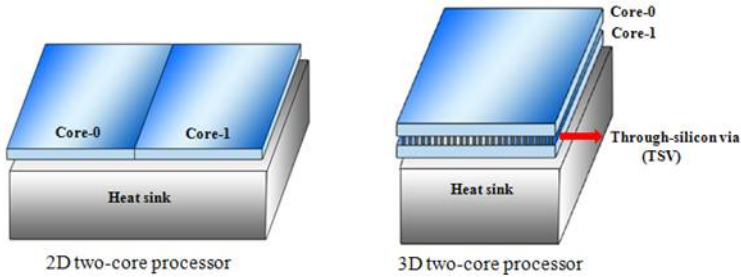


Fig. 1. Structural comparison between 2D two-core processor and 3D two-core processor

2.2 Dynamic Thermal Management Techniques

For the conventional 2D chips, dynamic thermal management(DTM) techniques have been widely used to address the thermal problems when the chip temperature exceeds the thermal limit supported by the cooling solutions[16]. A lot of researches have provided a large number of DTM techniques. They can be categorized into two different groups: One is hardware-based techniques and the other is software-based techniques. The hardware-based DTM techniques such as dynamic frequency scaling(DFS), dynamic voltage scaling(DVS) and clock gating[11] are more aggressive and effective in managing the temperature than the software-based techniques. However, the hardware-based techniques incur more execution-time overhead compared to the software-based techniques such as energy-aware process scheduling[17] and OS-level task scheduling[18].

The thermal problems are expected to be more severe in 3D multi-core processors compared to 2D multi-core processors. Therefore, more effective DTM techniques are required for addressing the thermal problems of 3D multi-core processors. The DFS technique has been regarded as one of the most efficient DTM techniques for 2D multi-core processors[19]. In this work, we analyze the detailed thermal behavior of 3D multi-core processors with the conventional DFS technique varying application features, cooling characteristics and frequency levels. We also propose an adaptive DFS technique to reduce the peak temperature of the 3D multi-core processor by assigning different DFS levels to each core depending on the corresponding cooling efficiency.

3 Adaptive Dynamic Frequency Scaling Technique

In the multi-core processor using the conventional DFS technique, the DFS levels applied to the cores have no difference because the target processor of the conventional DFS technique is the 2D multi-core processor where all the cores have comparable cooling efficiency. However, there is a significant difference in the cooling efficiency of the cores in the 3D multi-core processor depending on the vertical location of the core. For this reason, so as to mitigate the thermal problems in the 3D multi-core processor based on the fact that the cores with better cooling efficiency (the cores located nearer to the heat sink) can be clocked at a higher frequency compared to the cores with worse cooling efficiency[2], we propose an adaptive DFS technique which assigns different DFS levels to the cores by considering the corresponding cooling efficiency. We determine the DFS levels to the cores based on the theory in [20] to find the optimal frequency according to the temperature[21-22].

Table 1. DFS levels for the conventional DFS vs. DFS levels for the adaptive DFS

	Level-1	Level-2	Level-3	Level-4
core-0 (conventional DFS)	1GHz	2GHz	3GHz	4GHz
core-1 (conventional DFS)	1GHz	2GHz	3GHz	4GHz
core-0 (adaptive DFS)	0.5GHz	1.5GHz	2.5GHz	3.5GHz
core-1 (adaptive DFS)	1.5GHz	2.5GHz	3.5GHz	4.5GHz

The proposed adaptive DFS technique assigns different DFS levels to each core in the 3D multi-core processor by considering the cooling efficiency of the core. Therefore, in the 3D multi-core processor using the adaptive DFS technique, a higher baseline frequency is assigned to the core with better cooling efficiency while a lower baseline frequency is assigned to the core with worse cooling efficiency. In the 3D two-core processor depicted in Figure 1, the cooling efficiency of core-1 is better than that of core-0 owing to the shorter distance from the heat sink. For an example, as shown in Table 1, the proposed adaptive DFS technique assigns different DFS levels to the cores while the conventional DFS technique assigns same DFS levels to the cores. In the table, core-1 represents the core near the heat sink while core-0 means the core far from the heat sink. Based on the assumption that the conventional DFS has four DFS levels(1GHz, 2GHz, 3GHz, 4GHz), the higher four DFS levels(1.5GHz, 2.5GHz, 3.5GHz, 4.5GHz) are assigned to the core-1 in the adaptive DFS technique. To make the sum of the frequency to the same value, the lower four DFS levels(0.5GHz, 1.5GHz, 2.5GHz, 3.5GHz) are assigned to the core-0 in the adaptive DFS technique. As shown in Table 1, the sum of all the frequencies for the adaptive DFS is equal to that for the conventional DFS.

The baseline frequency of the core with relatively better cooling efficiency for the adaptive DFS is higher than that for the conventional DFS, while the baseline frequency of the core with relatively worse cooling efficiency for the adaptive DFS is lower than that for the conventional DFS. The frequency gap between the DFS levels for the adaptive DFS is equal to that for the conventional DFS.

We expect that the proposed adaptive DFS technique can reduce the peak temperature of the 3D multi-core processor compared to the conventional DFS, since the cooling efficiency of each core depending on the distance from the heat sink is considered in the proposed adaptive DFS technique.

4 Experiments

4.1 Experimental Methodology

Our baseline processor for the simulated core is Alpha21264(EV6)[23] without L2 cache, as shown in Figure 2. We extend it to a dual-core configuration for multi-core simulations. We use SimpleScalar [24] as our system simulator, which provides a detailed cycle-level modeling of processors. In addition, Wattch [25] is used to obtain a detailed power trace. We select two applications(mcf, gcc) from SPEC CPU2000 benchmark[26], because the chosen applications show a significant difference in the perspective of the peak temperature of the processor.

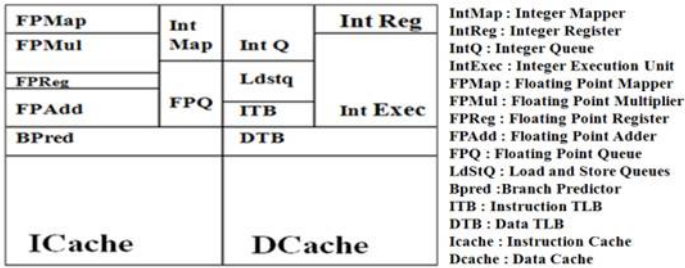


Fig. 2. Floorplan of Alpha21264(EV6)

HotSpot version 5.0 is used as our thermal modeling tool[27] and modeled parameters of the silicon layer and thermal interface material are shown in Table 2. Thermal modeling parameters can be obtained by considering material properties described in CRC handbook[28]. In the table, core-0 represents the core far from the heat sink, while core-1 is near the heat sink.

Table 2. Thermal modeling parameters

Parameter	Value			
	TIM-0	CORE-0	TIM-1	CORE-1
Specific heat capacity (J/m³K)	4.00e6	1.75e6	4.00e6	1.75e6
Thickness(m)	2.00e-5	1.50e-4	2.00e-5	1.50e-4
Resistivity (mK/W)	0.25	0.01	0.25	0.01

In the experiments, the DFS technique is engaged continuously instead of reacting to thermal emergencies. The conventional DFS technique is applied with four

levels(1GHz, 2GHz, 3GHz and 4GHz) and the sum of the frequencies of two cores is 5GHz for all simulations.

4.2 Thermal Impact of DFS on 3D Multi-core Processors

In order to analyze the thermal impact of DFS levels, the simulations are performed by running the same application(mcf) on both cores, only varying the DFS levels. Therefore, four schemes can be simulated as follows: 1GHz/4GHz, 2GHz/3GHz, 3GHz/2GHz, 4GHz/1GHz, where the slash separates the frequencies of both cores. The former represents the frequency of core-0 which is far from the heat sink and the latter refers to the frequency of core-1 which is near the heat sink.

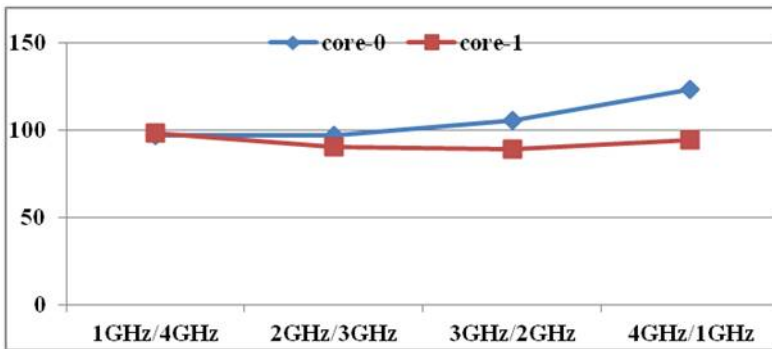


Fig. 3. Peak temperature(°C) according to DFS levels

The peak temperature according to DFS levels is shown in Figure 3. The peak temperature for the compared four schemes are 98.09°C(1GHz/4GHz), 97.17°C(2GHz/3GHz), 105.59°C(3GHz/2GHz) and 123.1°C(4GHz/1GHz). For the first scheme(1GHz/4GHz) and the last scheme(4GHz/1GHz), the same frequency values are used, just swapping the frequencies on the two cores, while the peak temperature changes from 98.09°C to 123.1°C. The same pattern can be seen in the second and the third scheme, the peak temperature varies from 97.17°C to 105.59°C. As shown in the graph, same frequency values yield a totally different thermal profile, because the cooling efficiency of each core in the 3D multi-core processor is not comparable. In general, the core near the heat sink shows better cooling efficiency than the core far from the heat sink. Therefore, to reduce the temperature of the hot unit in the core, the core with relatively better cooling efficiency should be clocked at a higher frequency than the core with relatively worse cooling efficiency.

4.3 Thermal Impact of Workload Distribution on 3D Multi-core Processors

In order to identify the thermal pattern according to the workload distribution for 3D multi-core processors, mcf and gcc applications are chosen to be run because these two benchmark applications show a great difference in generating thermal impact on

the processors. In the 2D single-core processor, mcf shows very high peak temperature(123.5°C), while gcc gets low peak temperature(89.9°C). In the experiments, two different applications are run on two cores with four DFS levels, resulting in eight different schemes as follows:

- mcf1GHz/gcc4GHz, mcf2GHz/gcc3GHz, mcf3GHz/gcc2GHz,
- mcf4GHz/gcc1GHz, gcc1GHz/mcf4GHz, gcc2GHz/mcf3GHz,
- gcc3GHz/mcf1GHz, gcc4GHz/mcf1GHz

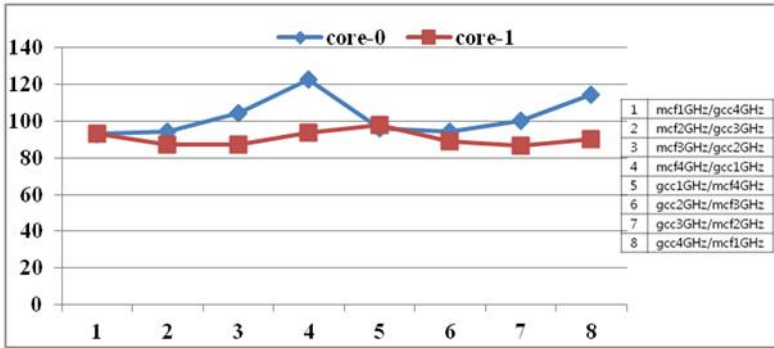


Fig. 4. Peak temperature(°C) according to workload distribution

The running application and DFS level of the core-0 are shown before the slash and those of the core-1 are shown after the slash. For the first four schemes, gcc is run on the core with better cooling efficiency and mcf is run on the other core. Then, applications are swapped for the last four schemes to observe the thermal impact of the workload distribution on 3D multi-core processors. As shown in Figure 4, the peak temperatures of eight simulated schemes are 93.08°C, 94.43°C, 104.1°C, 122.53°C, 97.51°C, 94.12°C, 100.01°C and 114.49°C, respectively. The core-1 yields lower temperature than the core-0 despite the type of workload because of its near-heat-sink location advantage(better cooling efficiency) except for fifth scheme(gcc1GHz/mcf4GHz). For the fifth scheme, the frequency of the core-1 is too higher than that of the core-0.

For the fourth scheme(mcf4GHz/gcc1GHz) and the eighth scheme(gcc4GHz/mcf1GHz), the frequency of each core is same, but the temperature of the eighth scheme is lower than that of the fourth scheme. Same pattern can be seen in the third and seventh scheme, reflecting the thermal impact of assigning different workloads to the cores with different cooling efficiency. Therefore, to reduce the temperature of the cores, the relatively heavier workload should be assigned to the core with relatively better cooling efficiency.

4.4 Conventional DFS Technique vs. Adaptive DFS Technique for 3D Multi-core Processors

In the processor using the conventional DFS technique, the DFS levels applied to each core are identical. However, our simulation results show that the core with better

cooling efficiency can be clocked at a higher frequency to mitigate the thermal problems in 3D multi-core processors. For this reason, the proposed adaptive DFS technique assigns different DFS levels to the cores depending on the corresponding cooling efficiency.

For a dual-core processor using the adaptive DFS technique, the DFS levels applied to the core with better cooling efficiency are 1.5GHz, 2.5GHz, 3.5GHz and 4.5GHz, while the DFS levels applied to the core with worse cooling efficiency are 0.5GHz, 1.5GHz, 2.5GHz and 3.5GHz. In the simulations, the sum of the frequencies of two cores in the adaptive DFS is set to 5GHz, because the sum of the frequencies of two cores is 5GHz in the conventional DFS. Therefore, for an example, one core is set to operate at 3.5GHz in the adaptive DFS, the frequency of the other core is set to 1.5GHz. Moreover, in order to make the results comparable with previous simulation results, same applications(mcf and gcc) are used. Simulated eight schemes for the adaptive DFS technique can be listed as follows:

mcf0.5GHz/gcc4.5GHz, mcf1.5GHz/gcc3.5GHz, mcf2.5GHz/gcc2.5GHz,
 mcf3.5GHz/gcc1.5GHz, gcc0.5GHz/mcf4.5GHz, gcc1.5GHz/mcf3.5GHz,
 gcc2.5GHz/mcf2.5GHz, gcc3.5GHz/mcf1.5GHz

The application and DFS level of the core-0 are shown before the slash and those of core-1 are shown after the slash. The peak temperature of the core in each scheme is shown in Figure 5.

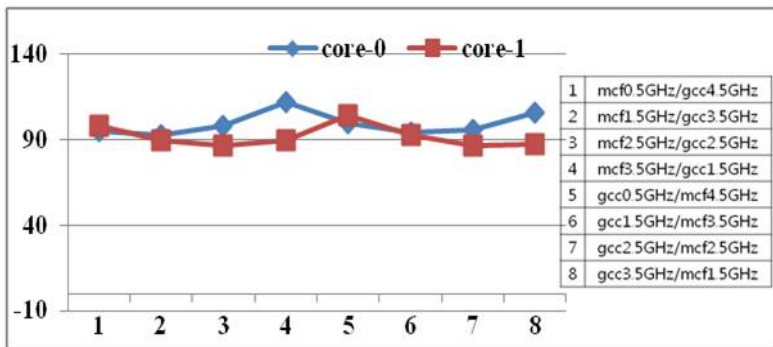


Fig. 5. Peak temperature(°C) with adaptive DFS

As shown in Figure 4 and Figure 5, the peak temperature with the adaptive DFS is lower than that with the conventional DFS except for the first scheme(mcf0.5GHz/gcc4.5GHz) and the fifth scheme(gcc0.5GHz/ mcf4.5GHz). The purpose of the adaptive DFS is reducing the frequency of the core with worse cooling efficiency while the frequency of the core with better cooling efficiency increases. In the simulated dual-core processor, the core with better cooling efficiency is core-1 while the core-0 has worse cooling efficiency. For the first scheme with the conventional DFS (shown in Figure 4), the peak temperature of core-1 is similar to that of core-0. For the fifth scheme with the conventional DFS, the peak temperature of core-1 is higher than that of core-0. For the first and fifth scheme with the adaptive DFS, the frequency of the

core-1 is higher than that with the conventional DFS. Therefore, the peak temperature of two schemes with the adaptive DFS scheme increases. However, the peak temperature with the adaptive DFS is lower than that with the conventional DFS by 5.01°C on the average. Especially, for the fourth scheme(gcc3.5GHz/ mcf1.5GHz) with the adaptive DFS, the peak temperature is lower than that with the conventional DFS by up to 10.35°C. Simulation results prove that assigning different DFS levels to each core depending on the cooling efficiency can reduce the peak temperature of the 3D multi-core processor, resulting in the improved reliability and better performance.

5 Conclusions

In this paper, we analyzed the thermal behavior of 3D multi-core processors varying DFS levels and workload distribution. We demonstrated that the core near the heat sink can be clocked at a higher frequency than the core far from the heat sink to keep the performance without thermal emergencies. The workload with bigger thermal influence also can be assigned to the core with better cooling efficiency to reduce the overall temperature of the 3D multi-core processor. We also proposed an adaptive DFS technique to mitigate the thermal problems in the 3D multi-core processor by assigning different DFS levels to each core based on the corresponding cooling efficiency. According to our simulation results, the proposed adaptive DFS technique reduces the peak temperature of the 3D multi-core processor by 5.01°C on the average compared to the conventional DFS technique. Therefore, we expect that the proposed adaptive DFS can be a good solution to reduce the peak temperature of the upcoming 3D multi-core processors.

Acknowledgements. This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2011-013-D00105).

References

1. Joyner, J.W., Zarkesh-Ha, P., Davis, J.A., Meindl, J.D.: A Three-Dimensional Stochastic Wire-Length Distribution for Variable Separation of Strata. In: Proc. of IEEE International Interconnect Technology Conference, San Francisco, USA, pp. 132–134 (June 2000)
2. Park, Y.J., Zeng, M., Lee, B.S., Lee, J.A., Kang, S.G., Kim, C.H.: Thermal Analysis for 3D Multi-core Processors with Dynamic Frequency Scaling. In: Proc. of IEEE/ACIS International Conference on Computer and Information Science, Kaminoyama, Japan, pp. 69–74 (August 2010)
3. Jang, H.B., Yoon, I., Kim, C.H., Shin, S., Chung, S.W.: The impact of liquid cooling on 3D multi-core processors. In: Proc. of the 2009 IEEE International Conference on Computer Design, California, USA, pp. 472–478 (October 2009)
4. Zhu, C., Gu, Z., Shang, L., Dick, R.P., Joseph, R.: Three-dimensional chip-multiprocessor run-time thermal management. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27(8), 1479–1492 (2008)

5. Puttaswamy, K., Loh, G.H.: Dynamic Instruction Schedulers in a 3-Dimensional Integration Technology. In: Proc. of the ACM Great Lake Symposium On VLSI, Philadelphia, USA, pp. 153–158 (May 2006)
6. Puttaswamy, K., Loh, G.H.: Implementing Caches in a 3D Technology for High Performance Processors. In: Proc. of the International Conference on Computer Design, San Jose, USA, pp. 525–532 (October 2005)
7. Reed, P., Yeung, G., Black, B.: Design Aspects of a Microprocessor Data Cache using 3D Die Interconnect Technology. In: Proc. of the International Conference on Integrated Circuit Design and Technology, pp. 15–18 (May 2005)
8. Puttaswamy, K., Loh, G.H.: Thermal Analysis of a 3D Die Stacked High Performance Microprocessor. In: Proc. of ACM Great Lakes Symposium on VLSI, Philadelphia, USA, pp. 19–24 (May 2006)
9. Davis, W.R., Wilson, J., Mick, S., Xu, J., Hua, H., Mineo, C., Sule, A.M., Steer, M., Franzon, P.D.: Demystifying 3D ICs: The Pros and Cons of Going Vertical. *IEEE Design Test Computers* 22, 498–510 (2005)
10. Cong, J., Luo, G.J., Wei, J., Zhang, Y.: Thermal-Aware 3D IC Placement Via Transformation. In: Proc. of ASP-DAC (2007), Yokohama, Japan, pp. 780–785 (January 2007)
11. Brooks, D., Martonosi, M.: Dynamic Thermal Management for High-performance Microprocessors. In: Proc. of the 7th International Symposium on High-Performance Computer Architecture, Monterrey, Mexico, pp. 171–182 (January 2001)
12. Coskun, A.K., Ayala, J.L., Atienza, D., Rosing, T.S., Leblebici, Y.: Dynamic Thermal Management in 3D Multicore Architectures. In: Proc. of Design, Automation & Test in Europe Conference & Exhibition, Nice, France, pp. 1410–1415 (April 2005)
13. Kumar, R., Zyuban, V., Tullsen, D.M.: Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling. In: Proc. of the 32th Annual International Symposium on Computer Architecture, Madison, USA, pp. 408–419 (June 2005)
14. Takahashi, K., Sekiguchi, M.: Through Silicon Via and 3-D Wafer/Chip Stacking Technology. In: Proc. of the 2006 Symposium on VLSI Circuit Digest of Technical Papers, Honolulu, USA, pp. 89–92 (June 2006)
15. Black, B., Annavaram, M., Brekelbaum, N., DeVale, J., Jiang, L., Loh, G.H., McCauley, D., Morrow, P., Nelson, D.W., Pantuso, D., Reed, P., Rupley, J., Shankar, S., Shen, J., Webb, C.: Die Stacking (3D) Microarchitecture. In: Proc. the 39th Annual IEEE/ACM International Symposium on Microarchitecture, Florida, USA, pp. 469–479 (December 2006)
16. Kumar, A., Shang, L., Peh, L.S., Jha, N.K.: System-level dynamic thermal management for high-performance microprocessors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27(1) (January 2008)
17. Mishra, R., Rastogi, N., Zhu, D., Mosse, D., Melhem, R.: Energy Aware Scheduling for Distributed Real-Time Systems. In: The Proc. International Parallel and Distributed Processing Symposium, Nice, France, pp. 21–30 (April 2003)
18. Zhou, X., Xu, Y., Du, Y., Zhang, Y., Yang, J.: Thermal Management for 3D Processors via Task Scheduling. In: Proc. of the 2008 37th International Conference on Parallel Processing, Portland, USA, pp. 115–122 (September 2008)
19. Skadron, K., Stan, M.R., Sankaranarayanan, K., Huang, W., Velusamy, S., Tarjan, D.: Temperature-aware microarchitecture: modeling and implementation. *ACM Transactions on Architecture and Code Optimization* 1(1), 94–125 (2004)

20. Shi, B., Zhang, Y., Srivastava, A.: Dynamic Thermal Management for Single and Multi-core Processors Under Soft Thermal Constraints. In: Proc. of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design, Poster, Austin, USA (August 2010)
21. Li, M., Zhao, W.: Visiting power laws in cyber-physical networking systems. *Mathematical Problems in Engineering* 2012, Article ID 302786, 13 pages (2012)
22. Li, M., Zhao, W.: Asymptotic identity in min-plus algebra: a report on CPNS. *Computational and Mathematical Methods in Medicine* 2012, Article ID 154038, 11 pages (2012)
23. Kessler, P.E.: The Alpha 21264 Microprocessor. *IEEE Micro* 19(2), 24–36 (1999)
24. Burger, D.C., Austin, T.M.: The SimpleScalar tool set, version 2.0. *ACM SIGARCH CAN* 25(3), 13–25 (1997)
25. Brooks, D., Tiwari, V., Martonosi, M.: Wattch: a framework for architectural-level power analysis and optimizations. In: Proc. of the 27th International Symposium on Computer Architecture, pp. 83–94 (June 2000)
26. Henning, J.L.: SPEC CPU 2000: Measuring CPU Performance in the New Millennium. *IEEE Computer* 33(7), 28–35 (2000)
27. Hotspot, <http://lava.cs.virginia.edu/HotSpot>
28. CRC Press, CRC Handbook of Chemistry, <http://www.hbcnetbase.com>

A Context-Aware Service Model Based on the OSGi Framework for u-Agricultural Environments

Jongsun Choi¹, Sangjoon Park², Jongchan Lee², and Yongyun Cho^{3,*}

¹ School of Computing, Soongsil University,
Sangdo 5 dong 1-1, Seoul, Korea
jongsun_choi@gmail.com

² Dept. of Computer Information Engineering, Kunsan National University,
1170 Daehangno, Gunsan, Jeonbuk, 573-701, Korea
lubimia@kunsan.ac.kr

³ Information and Communication Engineering, Suncheon National University,
413 Jungangno, Suncheon, Jeonnam 540-742, Korea
yycho@sunchon.ac.kr

Abstract. In ubiquitous environments, many services and devices may be heterogeneous and independent of each other. So, a service framework for ubiquitous environments have to be able to handle the heterogeneous applications and devices organically. Furthermore, because demand changes in ubiquitous environments tend to occur very dynamically and frequently, the service architecture also has to handle the demand changes of services in ubiquitous computing environments well. Services and devices in ubiquitous agricultural environments also have each other's different characteristics. Therefore, we need a service framework which can negotiate the heterogeneous characteristics of the services and devices for context-aware services in ubiquitous agricultural environments. In this paper, we propose an OSGi framework-based context-aware service model for ubiquitous agricultural environments. The proposed service model is based on OSGi framework, which can support various context-aware applications based on RFID/USN in ubiquitous agricultural environments regardless of what sensors and devices in agricultural environments are. Therefore, the proposed service model can fast reorganize and easily reuse existing service resources for a new agricultural service demand without all or big change of the established system architecture of the agricultural environments. Especially, the proposed service model can be greatly helpful in the developments of context-aware agricultural services in various cultivation environments equipped with each other's different sensors.

Keywords: context-aware service model, OSGi Framework, agricultural environments.

* Corresponding author.

1 Introduction

Recently, by using the newest information technologies, bio/organic sensor technologies and wire/wireless network technologies, agricultural environments have been smarter and more ubiquitous in the growth and development processes of crops in order to gain high productivity and safe growing of crops [1]. Commonly, services in ubiquitous agricultural environments tend to use various environmental conditions which related with cultivations of crops [1,2,3,4,5]. That is, a context-aware service in ubiquitous agricultural environments is based on various growth data gathered from RFID/USN. However, the devices and sensors distributed into the ubiquitous agricultural environments are heterogeneous and have different characteristics. So, the results from the sensors and the devices may mean various meanings.

This paper proposes an OSGi framework-based context-aware service model for ubiquitous agricultural environments. The proposed service model is based on OSGi framework [6], which has the valuable features such as plug-in architecture, open-type service bundles, system-independent service architecture, and so on. Therefore, the proposed service model is able to reorganize heterogeneous applications and devices in agricultural environments for a new context-aware service according to a user's demand, without any intervention during the execution of a service system. And, because the proposed service model can handle the demand changes of services in ubiquitous computing environments which tend to be changed dynamically and frequently, it is suitable for a context-aware agricultural service in ubiquitous agricultural environments which has to consider various agricultural environment conditions as important attributes for cultivation service executions. Therefore, with the suggested service model we can use it as a system-independent service architecture or a system-flexible service model to develop various context-aware service applications in agricultural environments.

The rest of this paper is organized as follows. In Section 2, we introduce related works. Section 3 describes the proposed service architecture. Finally, Section 4 concludes this paper.

2 Related Work

OSGi [6] is a set of specifications to offer easy reusing and convenient composing of service components having different characteristics. Figure 1 shows the OSGi's layered architecture. As shown Figure 1, OSGi contains a few of layers, which depict their roles. First, Bundles in Figure 1 is a layer to offer a service bundles which are installed on the OSGi framework by developers of service applications. Then, Services in Figure 1 is a layer for service application connected to the appropriate service bundles.

In this time, to connect a service application to a service bundle appropriated for it, OSGi offers a publish-find-bind method. Life-Cycle is a kind of APIs for bundles. OSGi framework executes on Java VM, and various components

based on OSGi can be implemented in Java. OSGi framework offers suitable architecture for the implementation of any service application which consists of heterogenous system components.

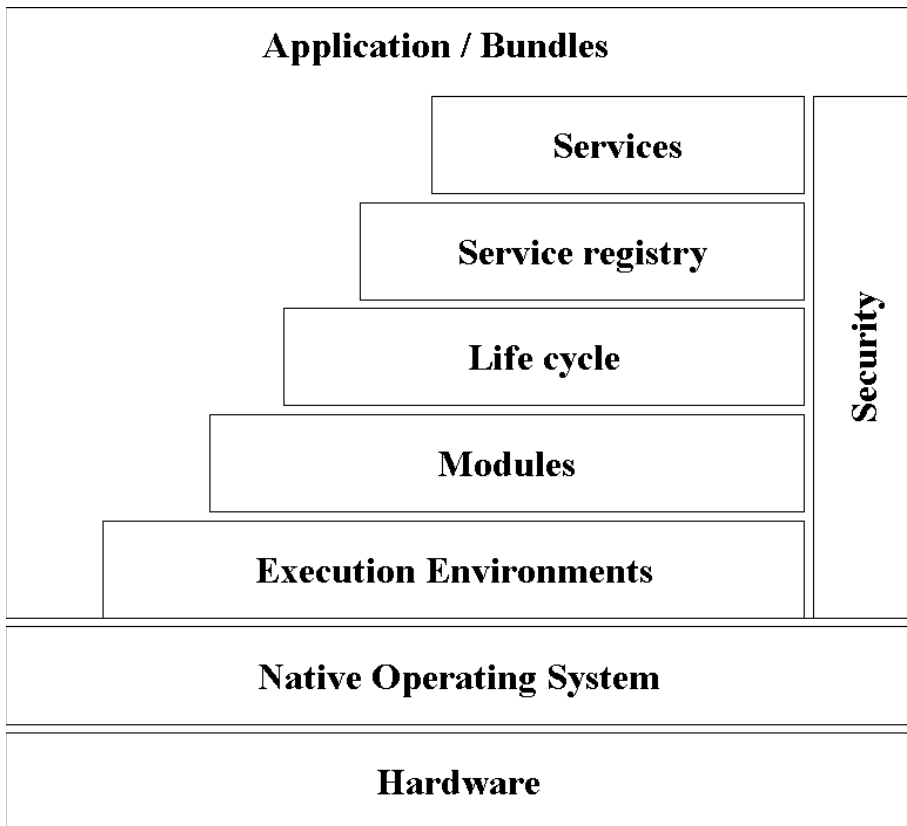


Fig. 1. The OSGi's layered architecture [6]

Until now, there have been many researches for context-aware service model and architecture based on OSGi [7,8,9,10,11,12]. FollowMe [7] is an OSGi-based context-aware workflow service framework that unifies a workflow-based application model and a context model using an ontology. FollowMe uses a scenario-based workflow model to dynamically handle user's service demands from various domains. Figure 2 illustrates the brief middleware architecture of FollowMe.

In Figure 2, the Physical layer is an area to deal with contexts from real sensors which may be many different from each others. The Device access layer is for mapping a physical sensor with any service by the sensor. The Platform layer contains OSGi framework to offer various OSGi-based service bundles and OSGi-based components. In the Platform layer, a context-aware service implemented

by a developer is executed. And, in the Application layer in Figure 2, a developer can compose or make a context-aware service or application.

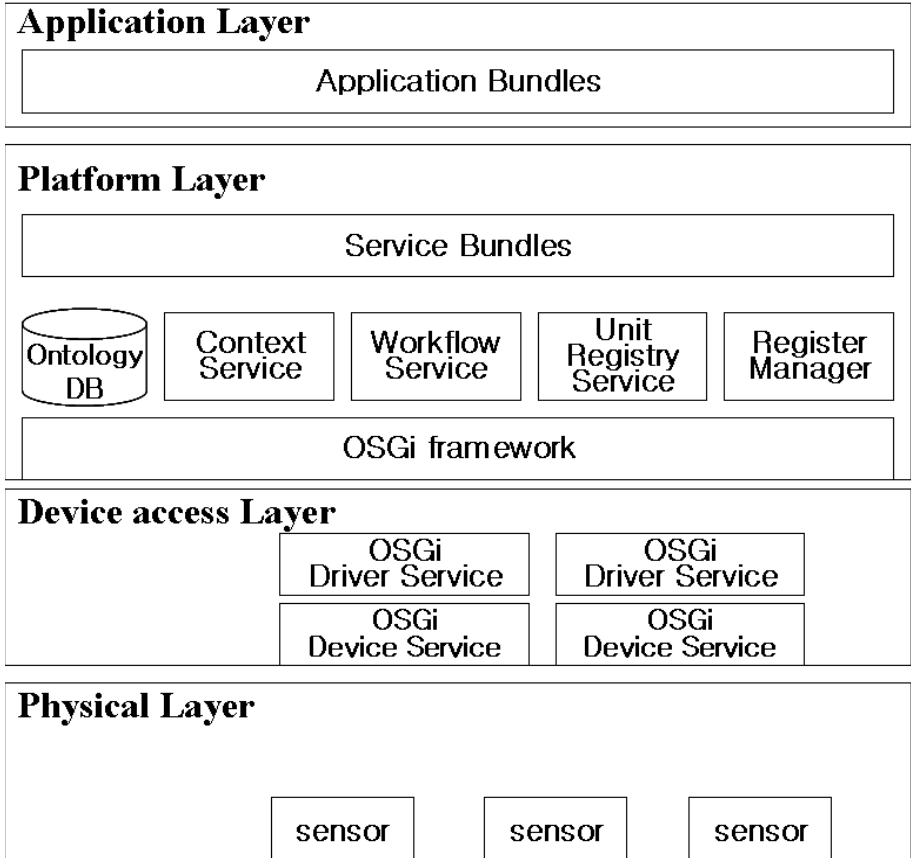


Fig. 2. The brief middleware architecture of FollowMe [7]

3 OSGi-Based Context-Aware Service Model for Ubiquitous Agricultural Environments

3.1 Layered Architecture of the Suggested Service Model

In this paper, we propose an OSGi framework-based context-aware service model for ubiquitous agricultural environments. To do this, the proposed service model consists of architectural layers to deal with various contexts based on real sensed data from different kinds of sensors, devices and hardware components. Figure 3 illustrates the layered architecture of the suggested service model for context-aware services in ubiquitous agricultural environments.

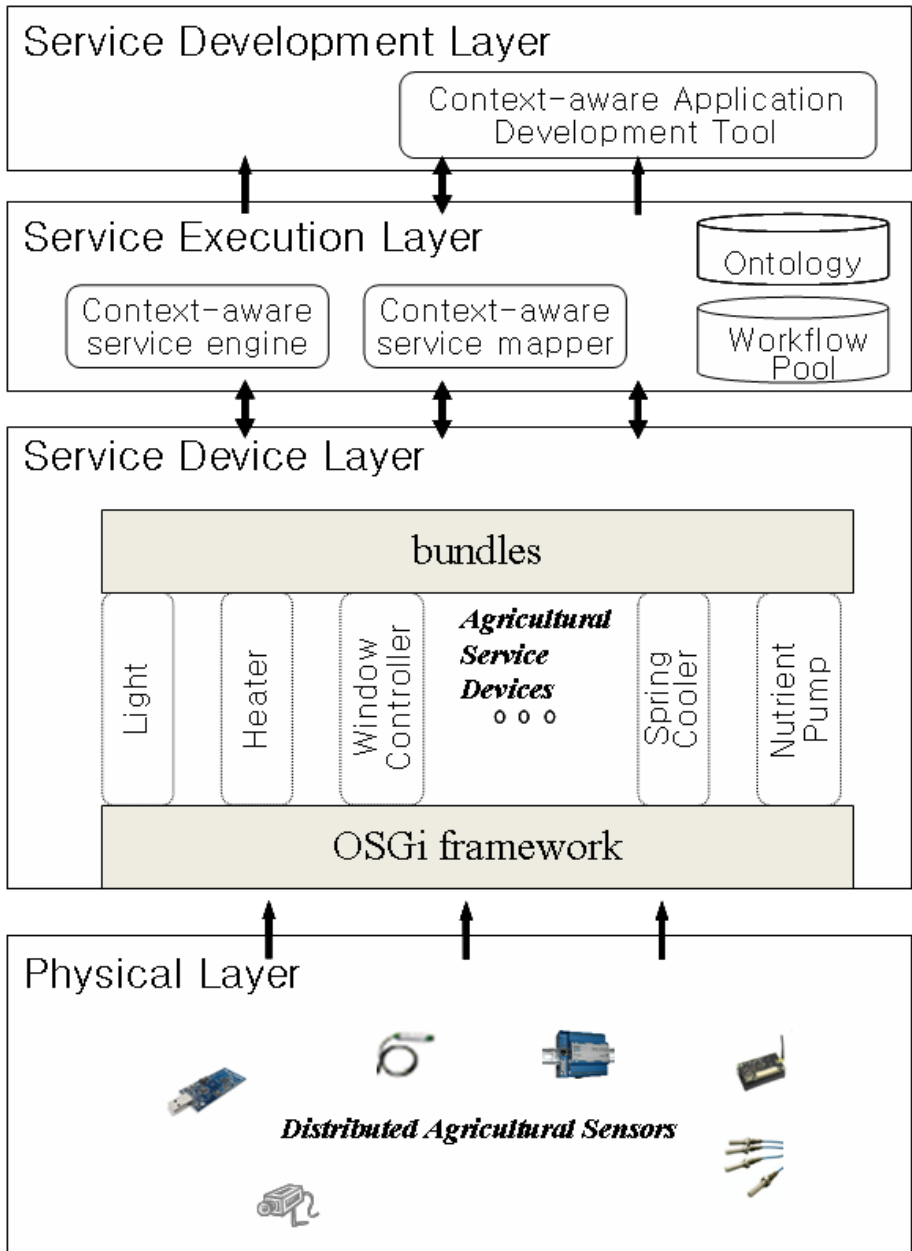


Fig. 3. The layered architecture of the suggested service model

As shown in Figure 3, the suggested service model's system architecture consists of four layers, which are a physical layer, a service device layer, a service execution layer, and a service development layer. Figure 4 shows a sample of a possible service scenario based on the proposed architecture for a context-aware agricultural service in ubiquitous agricultural environments.

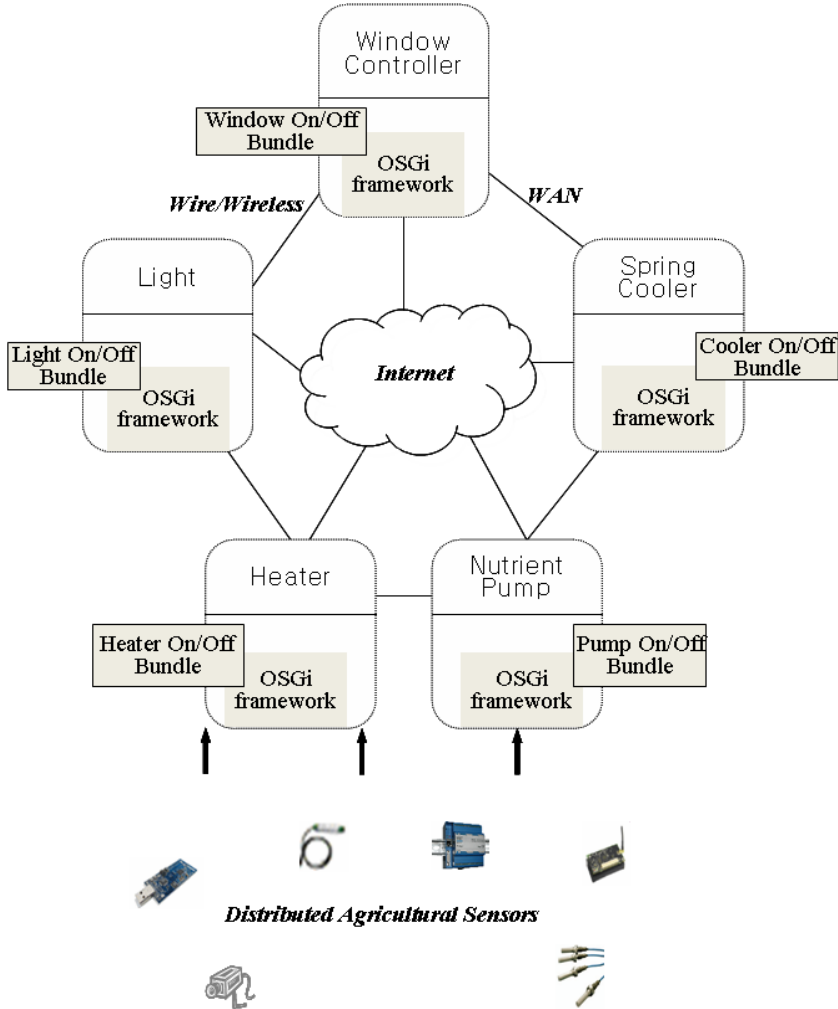


Fig. 4. A sample of a possible service scenario based on the proposed architecture for a context-aware agricultural service

As shown in Figure 4, the service devices includes own service device bundle based on OSGi framework. And the service devices can be connected to each other through Internet, wire/wireless networks, and WAN.

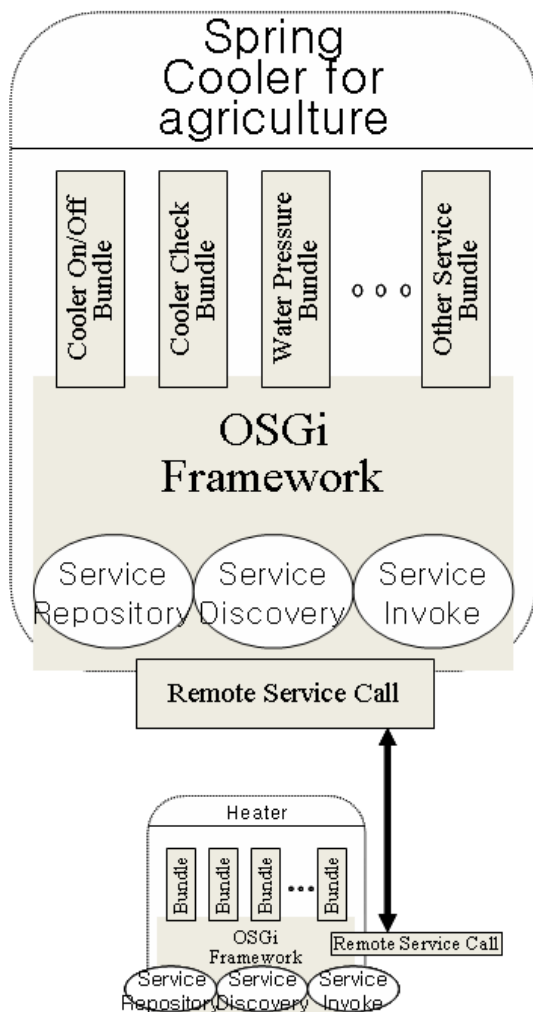


Fig. 5. More specific architecture of a service device for remote communication

Figure 5 shows more specific architecture of a service device to communicate data with any other remote service device. As shown in Figure 5, the architecture of the service device consists of various service device bundles implemented on OSGi framework.

Figure 6 shows a sample scenario to execute a context-aware agricultural service, when a humidity context related with crops is sensed from a humidity sensor in agricultural environments.

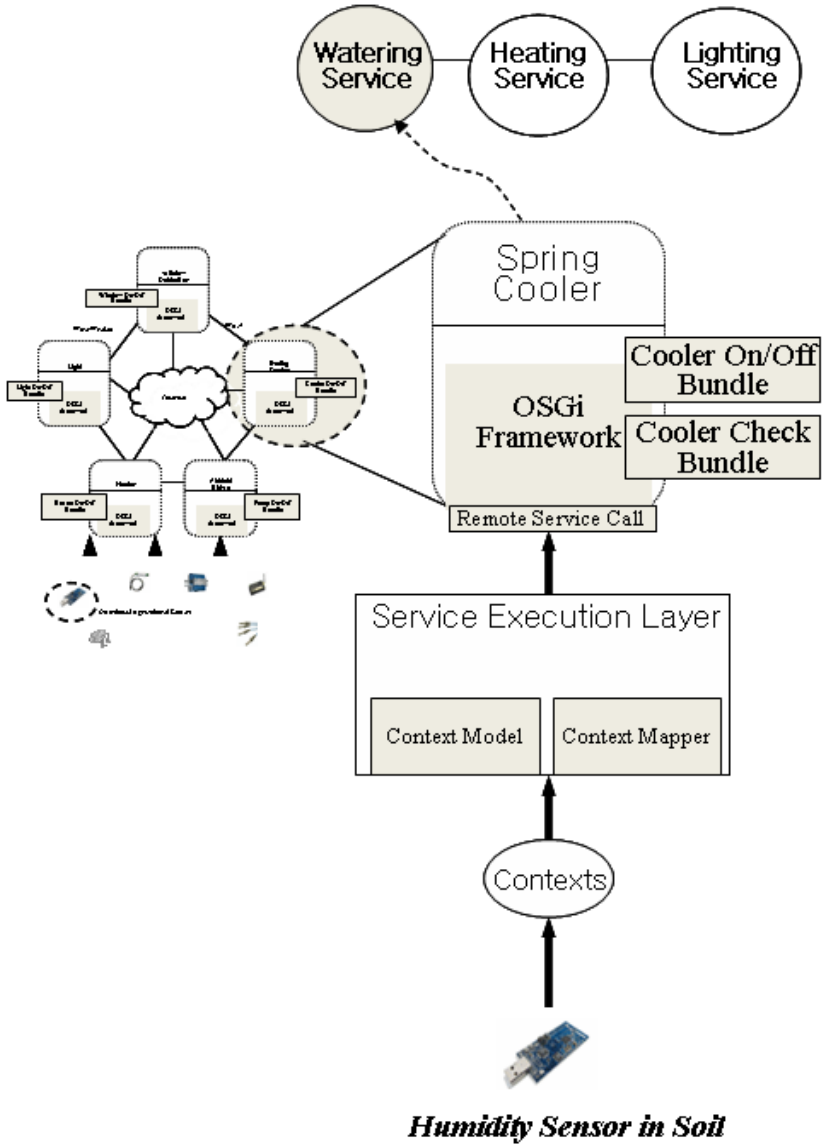


Fig. 6. A sample scenario to execute a context-aware agricultural service according to the context from real sensors

4 Conclusion

In this paper, we proposed an OSGi framework-based context-aware service model for ubiquitous agricultural environments. For this, we designed a layered system architecture based on OSGi framework, which supports plug-in architecture, open-

type service bundles, and system-independent service architecture. Therefore, the proposed service model is able to efficiently recompose different kinds of sensors and devices in agricultural environments in order to make a context-aware agricultural service. And, the proposed service model can easily reuse the heterogeneous resources according to the demand changes of services in ubiquitous computing environments. Therefore, with the suggested service model we can use it as a system-independent service architecture or a system-flexible service model to develop various context-aware service applications in agricultural environments. In future works, we are going to research the implementation of agricultural context-aware services based on the suggested OSGi framework.

References

1. Cho, Y., Yoe, H., Kim, H.: CAS4UA: A Context-Aware Service System Based on Workflow Model for Ubiquitous Agriculture. In: Kim, T.-H., Adeli, H. (eds.) AST/UCMA/ISA/ACN 2010. LNCS, vol. 6059, pp. 572–585. Springer, Heidelberg (2010)
2. Cho, Y., Park, S., Lee, J., Moon, J.: An OWL-Based Context Model for U-Agricultural Environments. In: Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., Apduhan, B.O. (eds.) ICCSA 2011, Part IV. LNCS, vol. 6785, pp. 452–461. Springer, Heidelberg (2011)
3. Cho, Y., Moon, J., Kim, I., Choi, J., Yoe, H.: Towards a smart service based on a context-aware workflow model in u-agriculture. *IJWGS* 7(2), 117–133 (2011)
4. Cho, Y., Yoe, H., Kim, H.: CAS4UA: A Context-Aware Service System Based on Workflow Model for Ubiquitous Agriculture. In: Kim, T.-h., Adeli, H. (eds.) AST/UCMA/ISA/ACN 2010. LNCS, vol. 6059, pp. 572–585. Springer, Heidelberg (2010)
5. Aqeel-ur-Rehman, Shaikh, Z.A.: Towards Design of Context-Aware Sensor Grid Framework for Agriculture. *World Academy of Science, Engineering and Technology* 38, 244–247 (2008)
6. The OSGi Alliance, <http://www.osgi.org/>
7. Li, J., Bu, Y., Chen, S., Tao, X., Lu, J.: FollowMe: On Research of Pluggable Infrastructure for Context-Awareness. In: *Proceedings of 20th International Conference on Advanced Information Networking and Applications (AINA 2006)*, vol. 1, pp. 199–204 (2006)
8. Gu, T., Pung, H.K., Zhang, D.Q.: Toward an OSGi-based infrastructure for context-aware applications. *IEEE Pervasive Computing* 3(4), 66–74 (2004)
9. Rellermeyer, J., Alonso, G., Roscoe, T.: R-OSGi: Distributed Applications Through Software Modularization. In: Cerqueira, R., Pasquale, F. (eds.) *Middleware 2007*. LNCS, vol. 4834, pp. 1–20. Springer, Heidelberg (2007)
10. Wu, J., Huang, L., Wang, D., Shen, F.: R-osgi-based architecture of distributed smart home system. *IEEE Transactions on Consumer Electronics* 54(3), 1166–1172 (2008)
11. Yu, Z., Zhou, Z., Yu, Z., Zhang, D., Chin, C.: An OSGi-based infrastructure for context-aware multimedia services. *IEEE Communications Magazine* 44(10), 136–142 (2006)
12. Lin, W., Sheng, Y.H.: Using OSGi UPnP and Zigbee to provide a wireless ubiquitous home healthcare environment. In: *The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, UBICOMM 2008*, pp. 268–273 (2008)

A Security Framework for Blocking New Types of Internet Worms in Ubiquitous Computing Environments

Iksu Kim¹ and Yongyun Cho^{2,*}

¹ School of Computing, Soongsil University,
Sangdo 5 dong 1-1, Seoul, Korea
iksuplicorer@gmail.com

² Information and Communication Engineering, Suncheon National University,
413 Jungangno, Suncheon, Jeonnam 540-742, Korea
yycho@sunchon.ac.kr

Abstract. In ubiquitous computing environments, many of services may communicate valuable data with each other through networking based on Internet. So, a service of ubiquitous computing environments has to be invulnerable to any type of Internet worms or attacks. Internet worms can compromise vulnerable servers in a short period of time. The security systems using signatures cannot protect servers from new types of Internet worms. In this paper, we propose a security framework for blocking new types of Internet worms, using the Snort and Honeytrap. Honeytrap are installed on every client, which detects Internet worm attacks. All interactions with Honeytrap are regarded as attacks because it is installed to lure Internet worms. In the proposed framework, when Honeytraps detect Internet worm attacks, Snort blocks the attacks.

Keywords: Internet worm, virtual machine, honeytrap.

1 Introduction

As the need for computer security increases, so does the need for computer security systems to be developed to provide adequate protection to all user. To improve computer security, security systems such as intrusion detection systems (IDSs) and intrusion prevention systems (IPSs) have been developed. Most security systems use the signatures of well-known attacks to detect intrusions and to protect computers from them. Therefore, it is important to collect information about new attacks because the detection rules employed by IDSs and IPSs are formulated using this information. Honeytraps are valuable security resources which are intended to get compromised. They aim to collect information regarding attackers. All activities within a honeypot are suspicious because it is not a production system. Honeytrap is a network security tool written to observe

* Corresponding author.

attacks against Internet services. It can relay incoming connections to a honeypot and at the same time record the whole communication. However, it is almost impossible to immediately generate detection rules from the information collected by honeypots.

In this paper, we propose a security framework for blocking Internet worm attacks, using Snort and Honeytrap. Honeytrap is installed on client computers to detect Internet worm attacks and Snort is installed on a firewall to blocking the attacks. All interactions with the honeytrap installed on client computers are regarded as attacks because the honeytrap is installed to lure Internet worms. Therefore, Honeytrap can detect the Internet worm attacks when an Internet worm broadcasts malicious packets to hosts in a network. Honeytrap then sends the source IP addresses of the attacks to a firewall and it can block the worm attacks.

The rest of this paper is organized as follows. Section 2 presents the background of information regarding attacks and previous studies. Section 3 describes the proposed framework. Finally, Section 4 concludes this paper.

2 Related Work

2.1 Detection Method for Attacks

A general method to detect attacks is to analyze log files. Unlike normal system logs, when attackers try to intrude a server, abnormal logs are recorded to log files. Figure 1 presents logs of port scans and a buffer overflow attack. Many connection logs are recorded in case of the port scans and unreadable characters are recorded in case of the buffer overflow attack. In order to detect these attacks, system administrators have to analyze log files periodically. Accordingly, it is hard to detect the attacks in real-time. For this reason, there have been many studies on IDSs.

```

Apr 14 19:18:56 victime in.telnetd[11634]: connect from xxx.168.11.200
Apr 14 19:18:56 victime imapd[11635]: connect from xxx.168.11.200
Apr 14 19:18:56 victime in.fingerd[11637]: connect from xxx.168.11.200
Apr 14 19:18:56 victime ipop3d[11638]: connect from xxx.168.11.200
Apr 14 19:18:56 victime in.telnetd[11639]: connect from xxx.168.11.200
Apr 14 19:18:56 victime in.ftpd[11640]: connect from xxx.168.11.200

Feb 23 08:19:29 ns rpc.statd[448]: gethostbyname error for ^X??X??Y??Y??Z??Z??[?]?
? ffff750 80497108052c20687465676 274736f6d616e797265206520726f7220726f66
bffff718 bffff719 bffff71a bffff71b ???? ???? ???? ???? ???? ???? ???? ???? ???? ????
???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ????
???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ????
???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ???? ????

```

Fig. 1. Logs of port scans and a buffer overflow attack

An IDS monitors a system and network and detects attacks in real-time. IDSs can be classified in network-based and host-based IDSs. To detect attacks, network-based IDSs monitor network traffic and host-based IDSs monitor and analyze log files, processes, and resources on the system. IDSs can also be classified into misuse-based and anomaly-based IDSs. Misuse-based IDSs use the signatures of known attacks to detect attacks. Anomaly-based IDSs establish a baseline of normal usage patterns and regard anomalous behavior as attacks.

RTSD (Real time scan detector) is a network-based and anomaly-based IDS to deal with illegal invasion attempts, and to try to detect the tactics, tendencies, and trends in illegal technologies [1]. It consists of the RTSD agent and the RTSD manager. The RTSD agent is run at various end sites and considers many connections from a host as port scanning. When the RTSD agent detects the port scanning, it logs the information and reports alerts to the RTSD manager. Accordingly, it is possible to collect statistics of scan incidents and to detect and identify previously unknown attacks. Song, J. and Kwon, Y. proposed a new real-time scan detection system that uses rules derived from attacker's behavioral pattern [2]. It can detect various attacks based on port scanning techniques and minimize the false positive rate.

Snort is an open source network-based and misuse-based IDS created by Roesch [3]. Through protocol analysis and content searching and matching, Snort can detect attack methods, including denial of service, buffer overflow, CGI attacks, stealth port scans, and SMB probes.

```
$EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS
(msg:"SHELLCODE Linux shellcode"; content:"\90 90 90 e8 c0 ff ff ff/bin/sh")
```

Fig. 2. Logs of port scans and a buffer overflow attack

Fig. 2 presents a rule to detect Linux shell code attacks. When Snort detects incoming packets with "90 90 90 e8 c0 ff ff ff/bin/sh" from an external network, it displays a message with "SHELLCODE Linux shellcode" on the screen. It is important to collect information of new attacks because IDSs use the detection rules created from the information. Unfortunately, traditional security systems, including firewalls, IDSs and others, collect vast amounts of data every day. The vast amounts of data make it difficult to find out new types of attacks.

A honeypot is a security resource whose value lies in being probed, attacked, or compromised [4]. All activities within a honeypot are suspicious because it is not a production system. The information collected by honeypots are highly valuable and provide white hats with a much fuller understanding of the intent, knowledge level, and modes of operation of attackers [5,6,7,8]. Honeypots have an advantage that they do not produce vast amounts of logs.

Honeytrap is a network security tool written to observe attacks against network services [9]. As a low-interactive honeypot, it collects information regarding known or unknown network-based attacks and thus can provide early-warning

information. When Honeytrap daemon detects a request to an unbound port, it considers the request as suspect. Honeytrap can relay incoming connections to honeypots and at the same time record the whole communication. However, it is impossible to immediately generate detection rules from the information collected by honeypots and Honeytrap.

2.2 A Change of Network Deployment for Security in Ubiquitous Computing Environments

Figure 3 shows a brief architecture of a traditional network deployment for security with Internet.

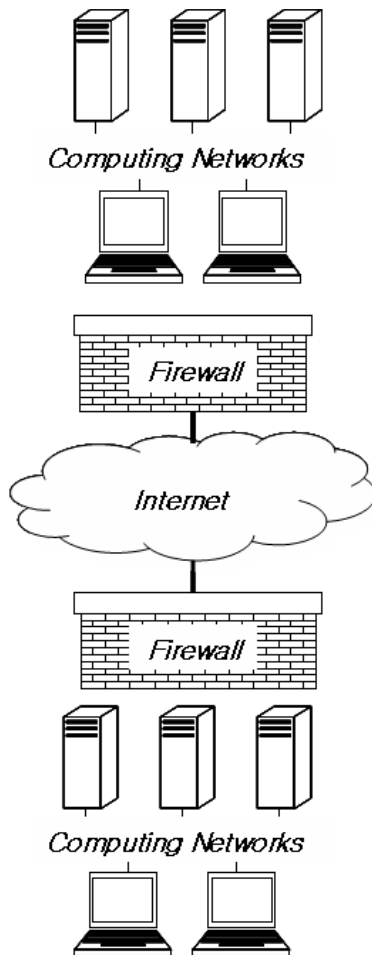


Fig. 3. A traditional network deployment for security with Internet

As shown in Figure 3, a network deployment for the traditional computing environments with the firewall can enough block various network attacks that may come into the computer systems through Internet. Figure 4 shows a network deployment for mobile Internet computing environments in which various mobile devices equipped with mobile Internet technologies are used.

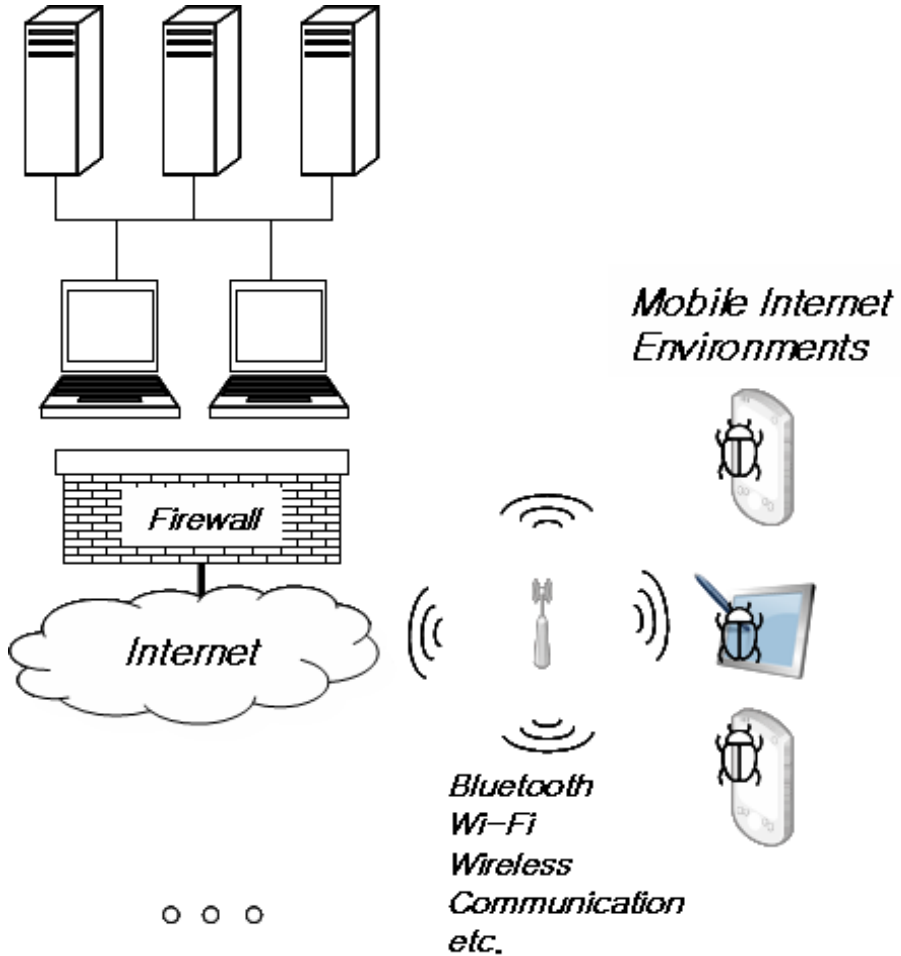


Fig. 4. A network deployment for security in mobile Internet environment through mobile devices

As shown in Figure 4, in ubiquitous computing environments, an user can freely access a network service with his/her hand-held devices with wireless Internet access technologies such as Wi-Fi, Wibro, and so on. So, a network

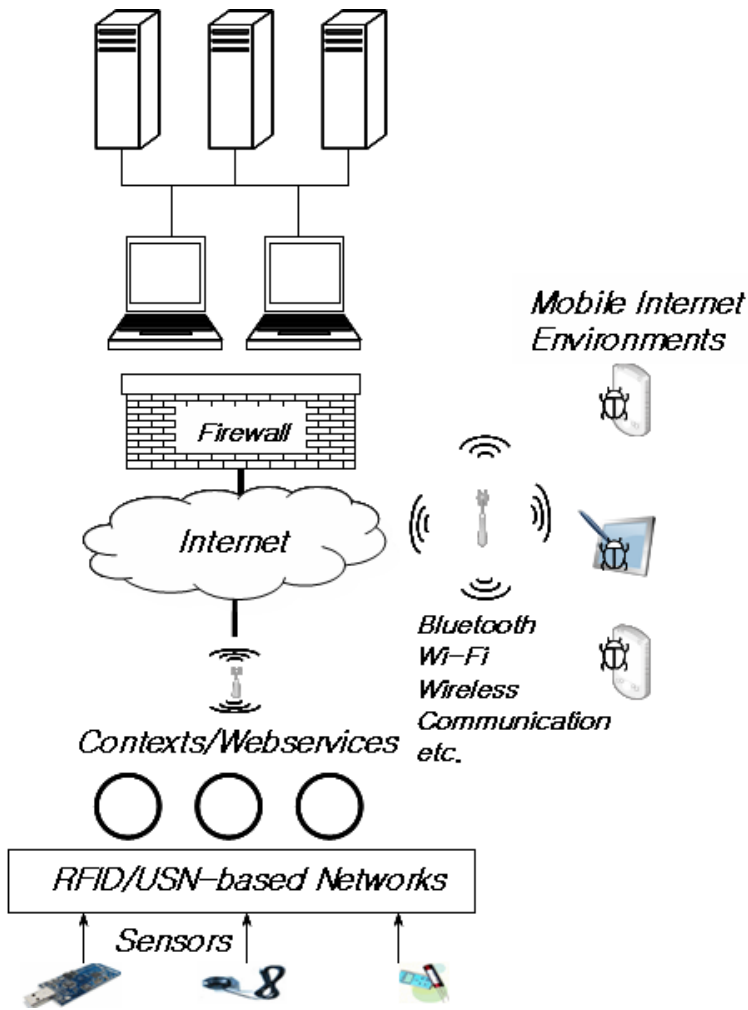


Fig. 5. A possible network deployment for security in ubiquitous computing environments with sensors and contexts

deployment for security in ubiquitous computing environments may be different from other computing environments. Figure 5 shows a possible network deployment for security in ubiquitous computing environments with various real sensors and contexts from the sensors through RFID/USN.

3 Security Framework for Blocking Internet Worm Attacks

3.1 Network Deployment of the Proposed Framework

As mentioned above, most security systems use the signatures of well-known attacks to detect and to block intrusions. However, Internet worms can compromise many vulnerable servers before creating rules for detecting the Internet worms. Figure 6 shows an example of network deployment using the proposed framework.

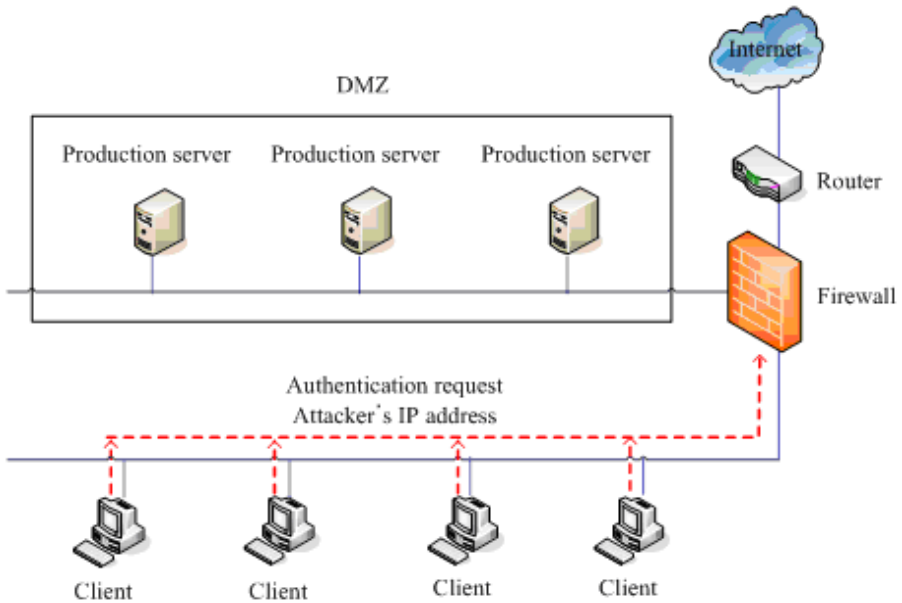


Fig. 6. An example of network deployment using the proposed framework

The goal of a proposed framework is promptly to detect and to block new types of Internet worms. In order to achieve the goal, virtual machines are deployed on every client computer with available unused IP addresses. All interactions with the virtual machines can be considered suspect interactions because the virtual machines are not used for services. After virtual machines are installed on client computers, Honeytraps are installed on the virtual machines. When an Internet worm broadcasts malicious packets to hosts in a network, client computers receive the packets. After that, the client computers inform an attacker's IP address to a firewall, as shown in Figure 6.

3.2 Components for Blocking Internet Worm Attacks

Client components for blocking internet worm attacks are shown in Figure 7. In the proposed framework, clients cannot use the Internet services without authentication. Therefore, clients have to send an authentication request to the firewall for using the Internet services. When a client executes Honeytrap, Authentication Requestor (AR) sends the authentication request to the firewall and the firewall generates the accept rule for the client.

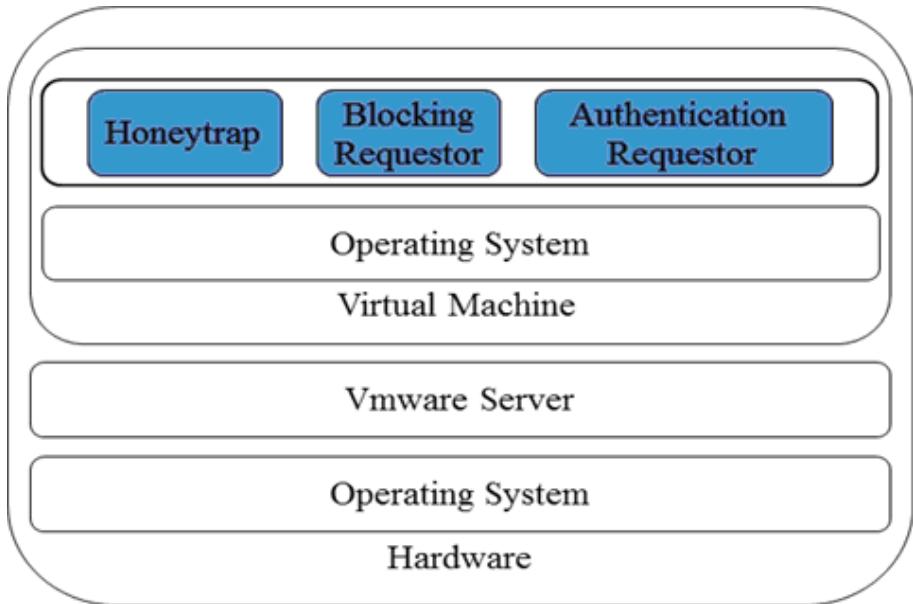


Fig. 7. Client components for blocking Internet worm attacks

Honeytrap opens ports on a client computer to detect malicious packets from Internet worms. The number of ports opened by Honeytrap is the same as the number of ports opened by production servers. When Honeytrap detects malicious packets from Internet worms, it records the packets and Blocking Requestor (BR) sends the source IP addresses of the packets to the firewall. The firewall then generates a drop rule for blocking the packets from the Internet worms.

Firewall components for blocking internet worm attacks are shown in Figure 8. After Receiver receives an authentication request message from AR, it hands over to Verifier. Verifier then generates an accept rule for a client. After that, the client can use Internet services. Snort can detect and block well-known Internet worm attacks with signature files, but it cannot detect new types of Internet

worms. In the proposed framework, when a new Internet worm broadcasts malicious packets to hosts in a network, Receiver re-receives the source IP address of the packets from BR. Receiver then generates a drop rule for blocking access from the IP address. From now on, all malicious packets from the IP address is blocked by Ip tables.

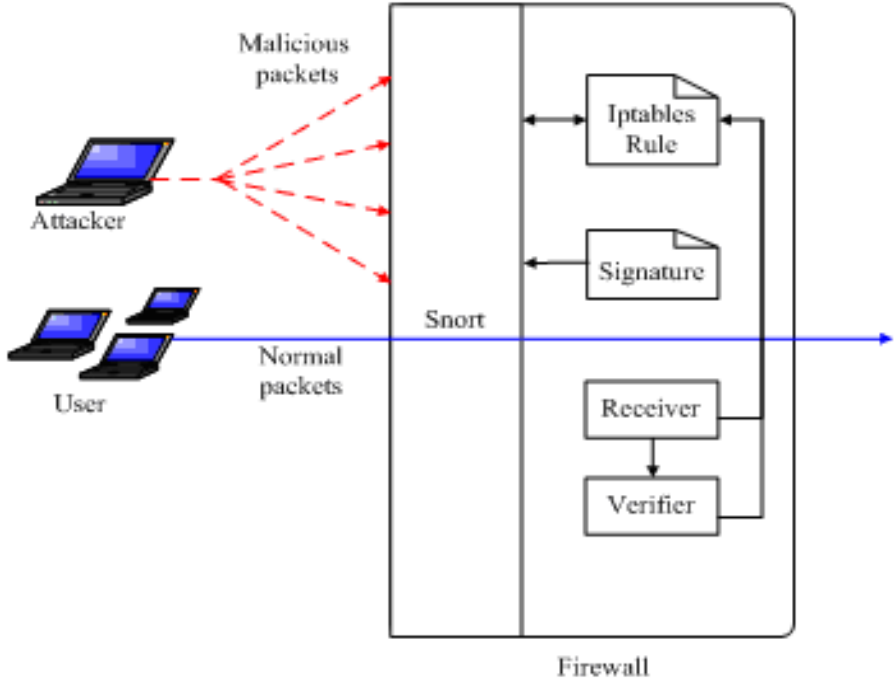


Fig. 8. Firewall components for blocking Internet worm attacks

4 Conclusion

Generally, IDS and IPS can detect well-known Internet worm attacks using signature files, but it cannot detect new Internet worm attacks. In order to solve the problem, we proposed a security framework for blocking new Internet worm attacks, using the Snort and Honeytrap. In the proposed framework, the firewall can block new Internet worms as well as well-known Internet worms due to Honeytrap.

References

1. Lee, H., Lee, S., Chung, H., Jeong, Y., Lim, C.: Analysis of Large Scale Network Vulnerability Scan Attacks and Implementation of the Scan-detection Tool. In: Proc. of the 11th Workshop on Information Security and Cryptography (1999)

2. Song, J., Kwon, Y.: An RTSD System against Various Attacks for Low False Positive Rate Based on Patterns of Attacker's Behaviors. *IEICE Transactions on Information and Systems* 89(10), 2637–2643 (2006)
3. Snort, <http://www.snort.org>
4. Spitzner, L.: *Honeypots: Tracking Hackers*. Addison-Wesley (2003)
5. Balas, E., Viecco, C.: Towards a Third Generation Data Capture Architecture for Honey-net. In: *Proc. of the 6th IEEE Information Assurance Workshop* (2005)
6. Dagon, D., Qin, X., Gu, G., Lee, W., Grizzard, J.B., Levine, J., Owen, H.: Honey-Stat: Local Worm Detection Using Honeypots. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) *RAID 2004*. LNCS, vol. 3224, pp. 39–58. Springer, Heidelberg (2004)
7. Levine, J., Grizzard, J., Owen, H.: Using Honeynets to Protect Large Enterprise networks. *IEEE Security and Privacy* 2(6), 73–75 (2004)
8. Pouget, F., Dacier, M.: Honeypot-based Forensics. In: *Proc. of AusCERT Asia Pacific Information Technology Security Conference* (2004)
9. Werner, T., <http://sourceforge.net/projects/honeytrap>

Quality Factors in Development Best Practices for Mobile Applications

Euler Horta Marinho^{1,2} and Rodolfo Ferreira Resende¹

¹ Computer Science Department, Universidade Federal de Minas Gerais
Belo Horizonte Minas Gerais 31270-010, Brazil

² Department of Exact and Applied Sciences, Universidade Federal de Ouro Preto
João Monlevade Minas Gerais 35931-008, Brazil
{eulerhm,rodolfo}@dcc.ufmg.br

Abstract. Smart mobile devices (hereafter, SMDs) are becoming pervasive and their applications have some particular attributes. Software Engineering deals with quality not only with traditional applications but also with process and product quality of this new application class. Models of software quality can aid to better understand the software characteristics that affect its quality. In this paper, we review some models of software quality factors, the best practices for SMD applications development proposed by UTI and W3C, and we discuss some of their relationships. We also discuss some deficiencies of the development best practices.

Keywords: software quality, development best practices, mobile applications, smart mobile devices.

1 Introduction

The software quality improvement has an important role in organizations involved with Information Technology. During the software life cycle, deficiencies in the quality of artifacts cause large scale costs [36].

The software quality definition must be made in order that the quality be measured in a meaningful way [27]. Measurements can be used to determine if the techniques improve the software and help understanding how process quality affects product quality [27]. Moreover, in development processes, the comprehension of quality characteristics should be associated with requirements specification and not only with the design and testing of an implemented system [11].

Several models of software quality factors have been proposed since 1970s [6]: McCall [28], Boehm [8], Dromey [15], FURPS [19] and, ISO 9126 [22], subsequently replaced by ISO 25010 [23] from the SQuaRE (Software product Quality Requirements and Evaluation) series of standards. As pointed out by Radulovic and García-Castro [32], models of software quality factors provide the basis for software evaluation and give a better insight of the software characteristics that

influence its quality. Furthermore, they ensure a consistent terminology for software product quality and provide guidance for its measurement. Stakeholders can perform measurements of software product quality aiming to make decisions related to software quality improvement, large-scale acquisitions, and contracts monitoring [25].

The smart mobile devices, for example, smartphones and tablets, are becoming pervasive. These devices are characterized by a wide range of interaction possibilities and some restrictions, which are not usually considered for non-portable computers. These aspects influence the use of SMDs and turn them distinct, but still competitive in relation to other computers.

With the SMDs, Software Engineering deals with the development of a new applications class [38]. SMDs applications (or mobile applications) have some particular attributes, for example a runtime environment with hardware constraints such as those related to low-power CPU, small memory, and small display [20]. Also, there may be constraints concerning lost connections, reduced bandwidth, and lack of network access [14]. As mentioned by Franke and Weise [17], these issues along with the diversity of platforms and strict time-to-market are one major reason for the lack of quality in SMDs software.

In this context, the Unified Testing Initiative (UTI), comprising organizations involved in the SMDs manufacturing and services (AT&T, Motorola, Oracle, and Samsung), drew up a guidance document containing development best practices with the aim to encourage adoption of practices to increase the quality of application for these devices [37]. Similarly, the World Wide Web Consortium (W3C) proposed a recommendation related to SMD web applications development. According to Charland and Leroux [9], the SMD applications development is based either on the programming of native applications or the implementation of sites and web services that interact with the device.

On the other hand, the literature presents some works analyzing models of software quality factors taking into account the peculiarities of applications for SMDs. Kim and Lee [26] used ISO 9126 in order to determine the most relevant software quality characteristics for mobile phones in the range of consumer electronics product. Franke and Weise [17] mentioned desirable attributes for a model of mobile software quality factors, some of these from ISO 9126 or Boehm's Model. Our work focus on the analysis of the software quality factors approached by development best practices suggested by the industry.

In this work, we present an evaluation of development best practices proposed by UTI and W3C regarding their adherence to quality factors from ISO 25010. We believe that our approach can help initiatives of extension and improvement of best practices and the quality improvement of the SMDs applications.

This paper is structured as follows. In Section 2, we discuss some models of software quality factors. In Section 3, we present the development best practices from UTI and W3C. In Section 4, we perform an evaluation of the development best practices for smart mobile devices applications. Section 5 presents the final considerations and future work.

2 Models of Software Quality Factors

According to the IEEE 1061 standard [21], the software quality is defined as “the degree to which software possesses a desired combination of quality attributes”. Also according to this standard, a quality factor is “a management-oriented attribute of software that contributes to its quality”. In the context of the requirements engineering, these attributes are associated to the non-functional requirements [18]. Thus, models of software quality factors can be used as frameworks for software quality specification and evaluation [12].

2.1 McCall’s Model

McCall and others proposed a model of software quality factors including 11 factors, in order to provide guidelines to the objective description of quality during the requirements specification phase. According to the authors, this allows the software quality measurement at all phases of the software development process. The model elements are shown in Table 1.

Table 1. Components of McCall’s Quality Model (Adapted from McCall and others [28])

Product Activity	Quality Factor	Quality Criteria
Product Operation	Correctness	Traceability, Consistency, Completeness
	Reliability	Error Tolerance, Consistency, Accuracy, Simplicity
	Efficiency	Execution Efficiency, Storage Efficiency
	Integrity	Access Control, Access Audit
	Usability	Training, Communicativeness, Operability
Product Revision	Maintainability	Consistency, Simplicity, Conciseness, Modularity, Self-Descriptiveness
	Flexibility	Modularity, Generality, Expandability, Self-Descriptiveness
	Testability	Simplicity, Modularity, Instrumentation, Self-Descriptiveness
Product Transition	Portability	Modularity, Self-Descriptiveness, Machine Independence, Software System, Independence
	Reusability	Generality, Modularity, Software System, Independence, Machine Independence, Self-Descriptiveness
	Interoperability	Modularity, Communications Commonality, Data Commonality

According to Table 1, the quality factors are organized in three groups that represent phases of product life cycle. Product Operation refers to the software’s

ability to be quickly understood, efficiently operated and able to provide the results required by the user. Product Revision is related to error correction and system adaptation [31]. Product Transition determines the software’s ability to be used in different software and hardware platforms, and to communicate with other systems. The established criteria can be associated with metrics and allow the determination of the relationships among the factors [28].

2.2 Boehm’s Model

Boehm and others proposed a model of software quality factors, where the quality factors are represented in a hierarchical manner. The high-level factors are associated with the software use, while the low-level factors are related to metrics. The model is shown in Figure 1.

The group of factors associated to *As-is Utility* determines that the software be reliable, efficient and has some usability aspects (*human-engineered*). According to Boehm and others [8], *As-is-Utility*, Maintainability and Portability are necessary, but not sufficient conditions to General Utility. This is because some domain specific softwares require additional factors not considered by the model (e.g. security).

2.3 FURPS Model

Robert Grady, while working at Hewlett-Packard, presented the FURPS model [19], whose acronym indicates the set of quality factors: Functionality, Usability, Reliability, Performance, and Supportability. The symbol ‘+’ was later added to the name of the model to include constraints related to design, implementation, interfaces and physical requirements. Table 2 shows the details of the FURPS model of software quality factors.

Table 2. Components of FURPS Model (Adapted from Grady [19])

Quality Factor	Description
Functionality	Feature Set, Capabilities, Generality, Security
Usability	Human factors, Aesthetics, Consistency, Documentation
Reliability	Frequency or Severity of failure, Recoverability, Predictability, Accuracy, Mean Time to Failure
Performance	Speed, Efficiency, Resource Consumption, Throughput, Response Time
Supportability	Testability, Extensibility, Adaptability, Maintainability, Compatibility, Configurability, Serviceability, Installability, Localizability, Portability

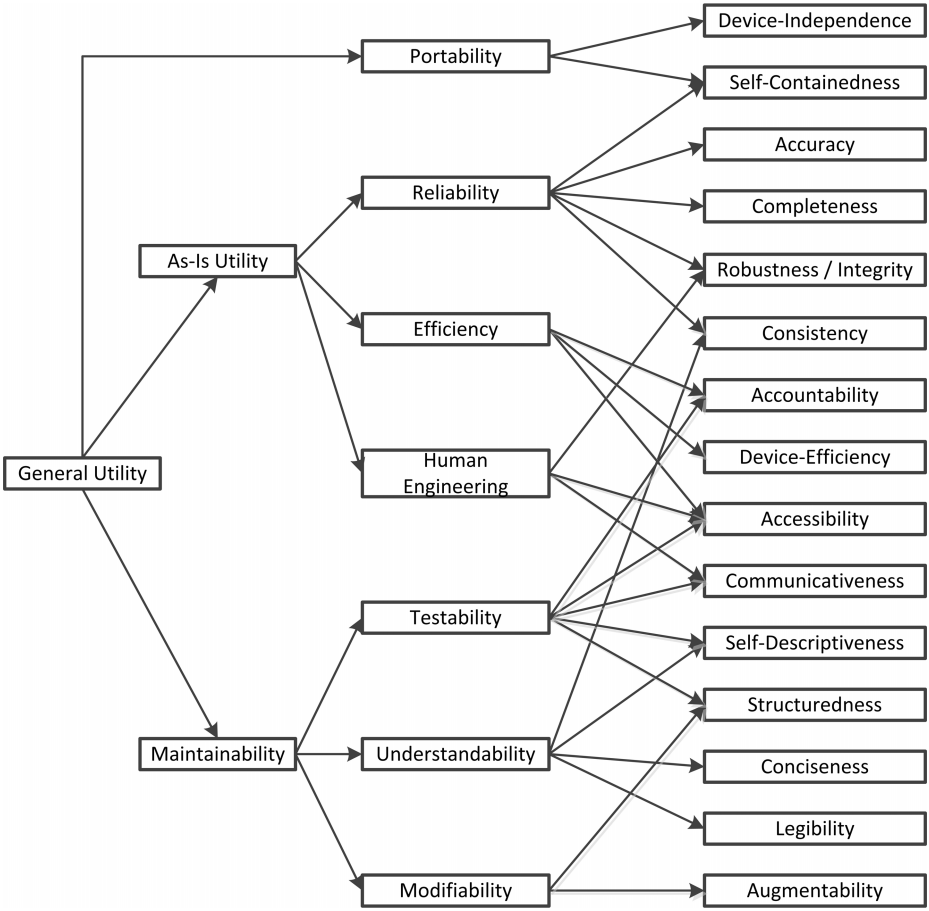


Fig. 1. Boehm's Model (Adapted from Boehm and others [8])

2.4 ISO 25010

ISO 9126 [22] is a standard that has two models of quality factors. The first gathers characteristics of internal and external quality, and the second, characteristics of quality in use. The model of internal and external quality factors describes how the product works in its development environment. The internal quality characteristics are related to the measurement of intermediate results, in terms of static aspects of artifacts related to the product. The external quality characteristics are related to behavioral aspects, measured from running the product code. The factors of quality in use model represent the vision of quality from the user perspective.

ISO 25010 [23], part of SQuARE series of standards, replaced ISO 9126. As part of the review process, the scope of the models of quality factors was extended to include computer systems, and quality in use from a system perspective. In this standard, the model of internal and external quality factors was named product quality model. Figure 2 illustrates this model with eight quality characteristics, each of them composed of a set of related subcharacteristics.

3 Development Best Practices for SMDs Applications

The UTI proposed development best practices (DBPs) for SMDs applications, which are platform-independent. The UTI's DBPs are organized taking account of several aspects of the use and operation of the application. Table 3 shows the description of the several aspects considered by UTI's DBPs.

Some DBPs can only be applied if there is application or hardware support. For example, the DBPs related to aspects 3 and 13, respectively.

These DBPs are not exhaustive and complete, for example, the security aspect can demand specific approaches to vulnerability detection [35,10]. DBPs to specific classes of systems, such as those with strict time constraints [24], are not considered. But then, these criteria have the backing of a large group of organizations.

W3C, standards organization for the Web, proposed a recommendation to aid the development of SMDs web applications. This recommendation has DBPs structured in a set of declarations considering several functional areas. Table 4 presents the functional areas descriptions of W3C's DBPs.

Similarly to the UTI's DBPs, some of W3C ones are dependent of the hardware support, for example the DBPs related to functional area 6. As noted by W3C [39], this recommendation does not have the aim to exhaust all the issues of functional area 2. However, it consider the most relevant aspects related to SMD web applications.

4 Evaluation of Development Best Practices for SMDs Applications

Preliminarily, we had to identify the set of quality factors in order to perform the evaluation. The models of software quality factors discussed in Section 2

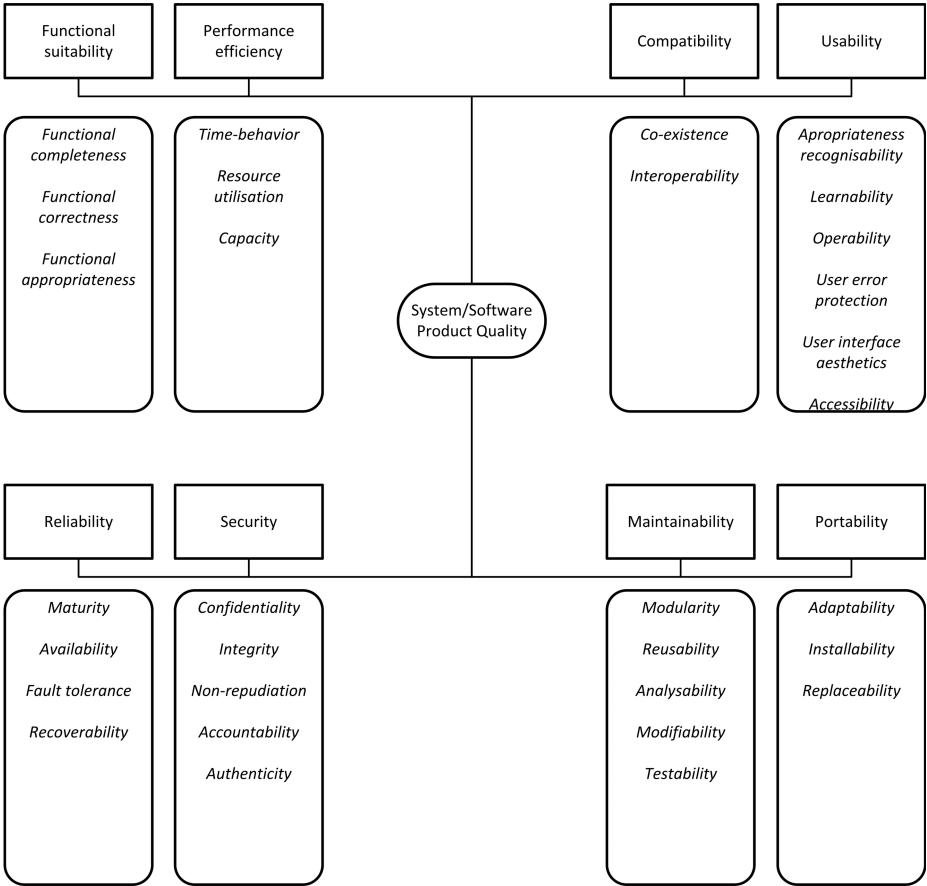


Fig. 2. Product Quality Model (Adapted from ISO 25010 [23])

Table 3. Aspects of the UTI's Best Practices

Aspect	Description	#DBPs
1. Installing and Launching	Characteristics of the life cycle of the application during the installing and launching stages	2
2. Memory and file storage	Exception handling during write operations to the file system	1
3. Connectivity	Handling of invalid or unusable network connections and connections with delays and loss of connectivity	4
4. Messaging and Calls	Characteristics of sending and receiving of text messages and handling of incoming calls	2
5. External Influence	Application behavior after memory card insertion	1
6. User Interface	Usability attributes of the application	17
7. Language	Support of local settings (e.g. date and time formats) and international characters	3
8. Performance	Attributes related to application performance including responsiveness and resource usage	5
9. Media	Attributes related to application settings	4
10. Menu	Use of standard menu items help and about	2
11. Functionality	Consistency of the implemented application functions with respect to their specifications	2
12. Keys	Correct and consistent use of the device keys	6
13. Device-specific Tests	State management in opening and closing actions of the device	1
14. Stability	State management in forced close by system and non-existence of crashes and freezes	2
15. Data Handling	Use of actions to save game state. Warning to user about deletion of data and availability of reversal of deletion	2
16. Security	Protection of sensitive user information	2

Table 4. Functional Areas of the W3C's Best Practices

Functional Area	Description	#DBPs
1. Application Data	Management of Web application's data	3
2. Security and Privacy	Protection of sensitive user information	1
3. User Awareness and Control	Control of application behavior not apparent	2
4. Conservative Use of Resources	Minimized use of device's resource	11
5. User Experience	Characteristics of overall user experience	10
6. Handling Variation in the Delivery Context	Treatment of application's content and navigation structure in according to delivery context (e.g. different device capabilities)	5

were analyzed and 20 factors were identified (Figure 3). All models refer to Reliability factor. Three models mention Usability and Portability factors. ISO 25010 identifies some factors from other models as quality subcharacteristics (for example, Integrity and Modifiability). On the other hand, this standard combines Performance and Efficiency factors as Performance efficiency. Thus, we adopted the ISO 25010 framework for evaluating the DBPs presented in Section 3, since its model of software quality factors has the necessary scope.

In order to determine the quality characteristics related to the DBPs, we used the following procedure:

1. Determine if the aspect (or the functional area) of the DBP is equivalent to quality characteristics or subcharacteristics directly (e.g. Security) or by similarity (e.g. Stability relates to Reliability, User Interface relates to Usability). In this case, the synonyms should be considered.
2. From the DBP description, check if it establishes constraints on the user experience (e.g. the system must show a progress bar), that suggests the Usability characteristic or constraints on the software's operation (e.g. the application must return to the same state before the interruption), that suggests the Reliability characteristic. For other quality characteristics or subcharacteristics, take in consideration the definitions in section 4.2 of ISO 25010 standard.

After applying the procedure, we performed a small scale validation of the results. Then, we obtained the distribution of quality characteristics depicted in Figure 4. A DBP may be related to more than one characteristic.

In accordance to Figure 4, the most frequent characteristic in the DBPs is Usability, followed by Performance efficiency, Portability, Reliability, and Compatibility. Portability, an important quality characteristic in SMDs applications [17], is addressed mainly by its subcharacteristic Adaptability. Functional suitability and Security are marginally approached by the DBPs. But then there is a

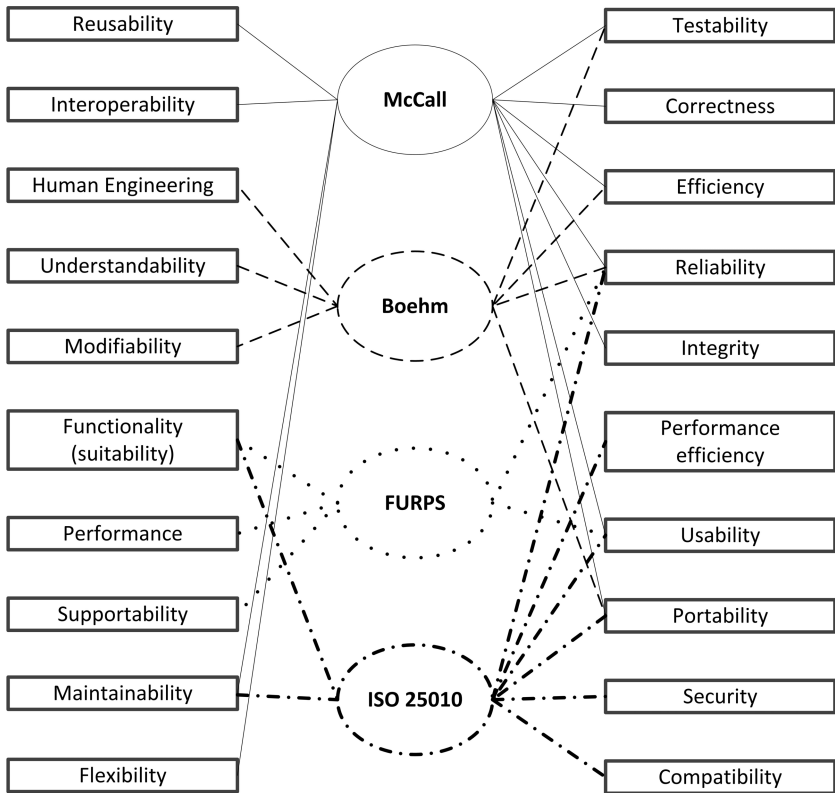


Fig. 3. Quality factors (Adapted from Samadhiya and others [33])

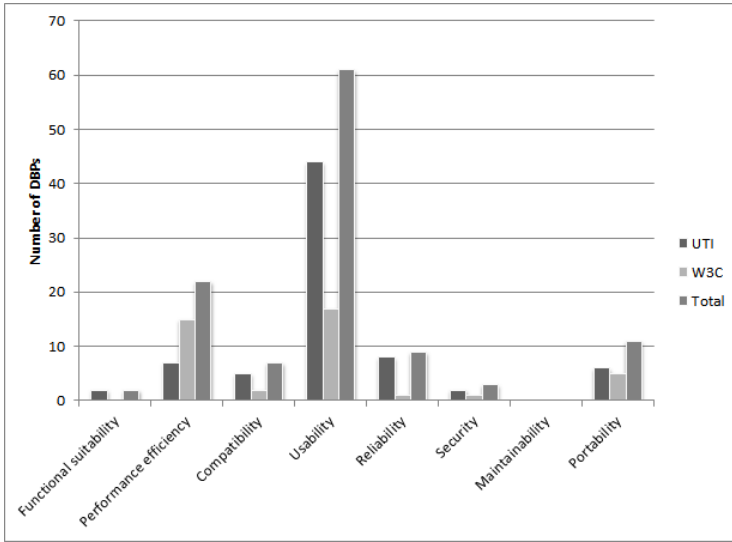


Fig. 4. Distribution of Quality Characteristics

lack of DBPs to deal with Maintainability, an important characteristic as noted by Kim and Lee [26].

Our results indicates that the DBPs are in conformity with the constraints of hardware environment of the SMDs applications mentioned in Section 1. Usability aspects of mobile applications are often different than those of other applications [5]. Also, this is the case of Performance efficiency [13]. On the other hand, for approaching the Security characteristic, it is strongly recommended the adoption of best practices throughout the software lifecycle [29,4]. Moreover, software maintenance best practices need to be recognized and better described [3].

During the evaluation, we noticed problems in the DBPs descriptions which are sometimes subjective. An example is the DBP definition related to memory card insertion while applications are running: “. . . the application should handle this gracefully” [37]. Also, UTI’s DBPs do not mention the existence of application specifications except those described in the help system. In spite of the argued role of help systems and user’s manuals as requirements specifications, they may leave out details to the validation of software requirements [7].

5 Conclusion

Software quality improvement is an important issue for the organizations involved with Information Technology. However, quality must be precisely defined in order to be measured [27]. In this context, models of software quality factors can offer a consistent terminology, provide insights about relationships among factors and instruct about their measurement [32].

Smart mobile devices are becoming pervasive and their applications have particular attributes. As mentioned by Hu and Neamtiu [20], these attributes and the developers' unfamiliarity with mobile platforms make SMDs applications prone to new kinds of bugs. An specific issue related to the SMD applications development is the application lifecycle management that is platform specific [16].

In this paper, we reviewed some models of software quality factors, the best practices for SMD applications development proposed by UTI and W3C, and we discussed some of their relationships. Our evaluation identified the most frequent factors in DBPs, taking into account the ISO 25010 quality factors. Moreover, we observed that DBPs exhibit deficiencies on the approach of important quality factors, such as Security and Maintainability. Despite the recommendation of the use of best practices throughout the software lifecycle [29,4] to deal with Security factor, for SMDs applications, it is necessary the use of techniques often different from those of non-portable computers [30]. Thus, we can realize the need for initiatives to extend and improve the development best practices for mobile applications.

As noted by Al-Kilidar and others [2], the generality of a standard such as ISO 9126 is inversely proportional to its utility. Thus, further investigations about the applicability of ISO 25010 are required, especially considering classes of applications such as those for SMDs which have technical and market specific demands [1,34].

We were able to perform a small scale validation of our approach and we are planning to perform a more comprehensive validation with a larger number of experts.

References

1. Abrahamsson, P., Hanhineva, A., Hullko, H., Ihme, T., Jaalinoja, J., Korkala, M., Koskela, J., Kyllonen, P., Salo, O.: Mobile-D: an agile approach for mobile application development. In: Companion of the 19th ACM SIGPLAN Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications, pp. 174–175. ACM, New York (2004)
2. Al-Kilidar, H., Cox, K., Kitchenham, B.: The use and usefulness of the ISO/IEC 9126 quality standard. In: International Symposium on Empirical Software Engineering, p. 7. IEEE, Los Alamitos (2005)
3. April, A., Abran, A.: A software maintenance maturity model (S3M): Measurement practices at maturity levels 3 and 4. *Electronic Notes in Theoretical Computer Science* 233, 73–87 (2009)
4. Ardi, S., Byers, D., Shahmehri, N.: Towards a structured unified process for software security. In: 2006 International Workshop on Software Engineering for Secure Systems, pp. 3–9. ACM, New York (2006)
5. Balagtas-Fernandez, F., Hussmann, H.: A methodology and framework to simplify usability analysis of mobile applications. In: IEEE/ACM International Conference on Automated Software Engineering, pp. 520–524. IEEE, Los Alamitos (2009)
6. Behkamal, B., Kahani, M., Akbari, M.K.: Customizing ISO 9126 quality model for evaluation of B2B applications. *Information and Software Technology* 51(3), 12–21 (2009)

7. Berry, D.M., Daudjee, K., Dong, J., Fainchtein, I., Nelson, M.A., Nelson, T., Ou, L.: User's manual as a requirements specification: Case studies. *Requirements Engineering* 9, 67–82 (2004)
8. Boehm, B.W., Brown, J.R., Lipow, M.: Quantitative evaluation of software quality. In: 2nd International Conference on Software Engineering, pp. 592–605. IEEE, Los Alamitos (1976)
9. Charland, A., Leroux, B.: Mobile application development: Web vs native. *Communications of the ACM* 54(5), 1–8 (2011)
10. Chin, E., Felt, A.P., Greenwood, K., Wagner, D.: Analyzing inter-application communication in Android. In: 9th International Conference on Mobile Systems, Applications, and Services, pp. 239–252. ACM, New York (2011)
11. Chung, L., do Prado Leite, J.C.S.: On Non-Functional Requirements in Software Engineering. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) *Conceptual Modeling: Foundations and Applications*. LNCS, vol. 5600, pp. 363–379. Springer, Heidelberg (2009)
12. Côté, M.A., Suryin, W., Georgiadou, E.: In search for a widely applicable and accepted software quality model for software quality engineering. *Software Quality Journal* 15(4), 401–416 (2007)
13. Díaz, A., Merino, P., Rivas, F.J.: Mobile application profiling for connected mobile devices. *IEEE Pervasive Computing* 9(1), 54–61 (2010)
14. Doernhoefer, M.: Surfing the net for software engineering notes. *SIGSOFT Software Engineering Notes* 35(5), 8–17 (2010)
15. Dromey, R.G.: A model for software product quality. *IEEE Transactions on Software Engineering* 21(2), 146–162 (1995)
16. Franke, D., Elsemann, C., Kowalewski, S., Weise, C.: Reverse engineering of mobile application lifecycles. In: 18th Working Conference on Reverse Engineering, pp. 283–292. IEEE, Los Alamitos (2011)
17. Franke, D., Weise, C.: Providing a software quality framework for testing of mobile applications. In: 4th IEEE International Conference on Software Testing, Verification, and Validation, pp. 431–434. IEEE, Los Alamitos (2011)
18. Glinz, M.: On non-functional requirements. In: 15th IEEE International Requirements Engineering Conference, pp. 21–26. IEEE, Los Alamitos (2007)
19. Grady, R.B.: *Practical Software Metrics for Project Management and Process Improvement*. Prentice Hall, Englewood Cliffs (1992)
20. Hu, C., Neamtiu, I.: Automating GUI testing for android applications. In: 6th IEEE/ACM International Workshop on Automation of Software Test, pp. 77–83. ACM, New York (2011)
21. IEEE: IEEE Std 1061-1998: IEEE Standard for a Software Quality Metrics Methodology (1998)
22. ISO: ISO/IEC 9126-1:2001, Software Engineering - Product Quality - Part1: Quality Model (2001)
23. ISO: ISO/IEC 25010:2011, Systems and Software Engineering - Systems and software Quality Requirements and Evaluation (SQuARE) - System and Software Quality Models (2011)
24. Jeong, K., Moon, H.: Object detection using FAST corner detector based on smartphone platforms. In: 1st ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering, pp. 111–115. IEEE, Los Alamitos (2011)
25. Jung, H.W., Kim, S.G., Chung, C.S.: Measuring software product quality: A survey of ISO/IEC 9126. *IEEE Software* 21(5), 88–92 (2004)

26. Kim, C., Lee, K.: Software quality model for consumer electronics product. In: 9th International Conference on Quality Software, pp. 390–395. IEEE, Los Alamitos (2009)
27. Kitchenham, B., Pfleeger, S.L.: Software quality: The elusive target. *IEEE Software* 13(1), 12–21 (1996)
28. McCall, J.A., Richards, P.K., Walters, G.F.: *Factors in Software Quality*, vol. 1-3. Nat'l Tech. Information Service, Springfield, USA (1977)
29. Mead, N.R., McGraw, G.: A portal for software security. *IEEE Security & Privacy* 3(4), 75–79 (2005)
30. Oberheide, J., Jahanian, F.: When mobile is harder than fixed (and vice versa): demystifying security challenges in mobile environments. In: 11th Workshop on Mobile Computing Systems & Applications, pp. 43–48. ACM, New York (2010)
31. Ortega, M., Pérez, M., Rojas, T.: Construction of a systemic quality model for evaluating a software product. *Software Quality Journal* 11(4), 219–242 (2003)
32. Radulovic, F., García-Castro, R.: Towards a Quality Model for Semantic Technologies. In: Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., Apduhan, B.O. (eds.) ICCSA 2011, Part V. LNCS, vol. 6786, pp. 244–256. Springer, Heidelberg (2011)
33. Samadhiya, D., Wang, S.H., Chen, D.: Quality models: Role and value in software engineering. In: 2nd International Conference on Software Technology and Engineering, pp. V1–320–V1–324. IEEE, Los Alamitos (2010)
34. Scharf, C., Verma, R.: Scrum to support mobile application development projects in a just-in-time learning context. In: 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, pp. 25–31. ACM, New York (2010)
35. Shabtai, A., Fledel, Y., Kanonov, U., Elovici, Y., Dolev, S., Glezer, C.: Google Android: a comprehensive security assesment. *IEEE Security & Privacy* 8(2), 35–44 (2010)
36. Streit, J., Pizka, M.: Why software quality improvement fails (and how to succeed nevertheless). In: 33rd International Conference on Software Engineering, pp. 726–735. IEEE, Los Alamitos (2011)
37. Unified Testing Initiative, The: Best Practice Guidelines for Developing Quality Mobile Applications, http://www.unifiedtestinginitiative.org/files/uti_best_practices_v1_final.pdf (last visited December 19, 2011)
38. Wasserman, A.I.: Software engineering issues for mobile application development. In: FSE/SDP Workshop on the Future of Software Engineering Research, pp. 397–400. ACM, New York (2010)
39. World Wide Web Consortium, The: Mobile Web Applications Best Practices (December 14, 2010), <http://www.w3.org/TR/mwabp/>, (last visited December 19, 2011)

ShadowNet: An Active Defense Infrastructure for Insider Cyber Attack Prevention

Xiaohui Cui¹, Wade Gasior², Justin Beaver¹, and Jim Treadwell¹

¹ Oak Ridge National Laboratory
Oak Ridge, TN, USA

{cuix,beaverjm,treadwelljn}@ornl.gov
<http://cda.ornl.gov>

² University of Tennessee at Chattanooga
Chattanooga, TN, USA
wadegasior@gmail.com

Abstract. The ShadowNet infrastructure for insider cyber attack prevention is comprised of a tiered server system that is able to dynamically redirect dangerous/suspicious network traffic away from production servers that provide web, ftp, database and other vital services to cloned virtual machines in a quarantined environment. This is done transparently from the point of view of both the attacker and normal users. Existing connections, such as SSH sessions, are not interrupted. Any malicious activity performed by the attacker on a quarantined server is not reflected on the production server. The attacker is provided services from the quarantined server, which creates the impression that the attacks performed are successful. The activities of the attacker on the quarantined system are able to be recorded much like a honeypot system for forensic analysis.

1 Introduction

Cyber security has become a national priority. Despite the number of recent news reports about hacker attacks and external network intrusions, trusted employees and business partners with authorized access to network still pose the greatest security risk to the government and private companies [1]. It is usually assumed that users who are given access to network resources can be trusted. However, the eighth annual CSI/FBI 2003 report [2] found that insider abuse of network access was the most cited form of attack or abuse. It is reported that 80 percent of respondents were concerned about insider abuse, although 92 percent of the responding organizations employed some form of access control and insider prevention mechanism. There are also large amount of insiders committing espionage cases have caused tremendous damage to U.S. national security. Two infamous insider threat cases, one is the case of former FBI agent R. P. Hanssen, who was convicted for spying for Russia. The another case is the United States diplomatic cables leak. In the FBI Hanssen case, over a span of more than 15 years, Hanssen provided his Russian contacts with highly classified

documents and details about U.S. intelligence sources and electronic surveillance taken directly from his employer, the FBI.

Detecting and preventing insider user misuse involves many challenges because insiders understand their organization's computer system and how the computer network system works. Inside users typically also have greater knowledge than outsiders do about system vulnerabilities. Therefore, the chances of a successful attack can be greater for an insider attack than for an outsider attack [4]. For instance, the knowledge that a malicious insider has about the sensitivity of information gives him/her a better chance to breach information confidentiality. Insider user misuse is different from outsider misuse with respect to the nature of the threats that both cause. However, because of lacking of understanding the differences in implementation of detection and prevention techniques between insider misuse and outsider misuse, most institutions intent to apply existing cyber security techniques to both threats [5].

2 Related work

A Honeypot [6,7] is an internet-attached server that acts as a decoy, luring in potential hackers in order to study their activities and monitor how they are able to break into a system [8]. As shown in Fig.1, the traditional Honeypots are designed to mimic systems that an intruder would like to break into, but limit the intruder from having access to an entire network. The most widely used honeypots is the honeyds [9] that run on a honeyd server and represent unused IP addresses in the organization's network. They function through emulating operating systems and services, thus allowing them to interact with the attacker. Any attempted connection to one of the honeypot servers is assumed to be unauthorized (malicious activity). An outside attacker most likely has little knowledge about the enterprise network structure and the network location (Internal IP address) of the sensitive information stored. Those outsiders have to depend on NMAP [10] or other network scanning software for mapping the target network structure and finding out the most vulnerable system for hacking. It is possible that an intruder might spend days hacking into an old Windows system that is only used as printer server and contains no valuable information. This kind of situation gives the traditional honeypot technology a chance for acting as a decoy vulnerable system and attracting intruder attacks. By using honeypot in a network, it can reduce the change for intruder find out the real valuable target.

However, the traditional honeypot may have difficulty in preventing attacker who has some insider information about the network. Different from outsider, the insider has more information about the enterprize network architecture and the computer system he wants to attack. The malicious insider most likely knows the IP address or machine name where the sensitive information is stored. As an employee of the enterprize, the intruder can easily access the enterprize internal network without passing through the enterprize firewall. So, an insider doesn't need to use NMAP or other network scan software to randomly discover

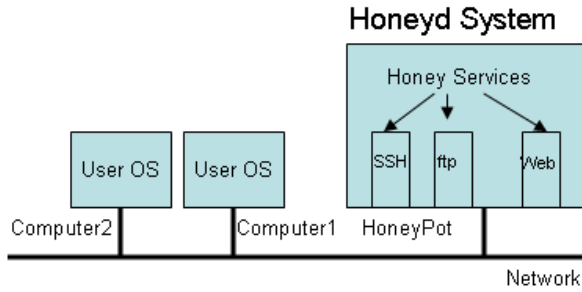


Fig. 1. Traditional HoneyPot Infrastructure

the vulnerable system in the network. Instead, the valuable system location, IP address even low privilege user account can be easily discovered by insider by using social engineering. To protect from being exposed, some insiders might not steal valuable data that he can legally access. Instead, they will use hacking tools to acquire information that he can not legally access. Before the insider launch an attack, he already knows the IP address of the systems that may have valuable data and he may even have low-level privileges to legally access the system. In this case, deploying the traditional honeypot system will not help detect a malicious insider.

3 An Active Defence Infrastructure

3.1 Overview

In this research, we developed an active defense infrastructure, called ShadowNet, for insider attack prevention and forensic analysis. The diagram of the infrastructure is shown in Fig. 2 and Fig. 3. The purpose of this ShadowNet is to help mitigate risk in an organization by actively preventing the malicious insider to harm a real computer or application and preventing him from spreading his attack to other computing resources. At the same time, the system provides a mechanism for real time collecting forensic data without the risk of shutting down the attacked system or leaking any stored sensitive information.

This infrastructure includes two elements: the ShadowNet client and the ShadowNet server. Different from the traditional honeypot strategy where static honeypot servers are placed in the network to lure attackers, the developed infrastructure will actively deploy a decoy host system only when host is under attack or suspicious behaviors are noticed. This capability is achieved by engaging the attacker with a virtual live clone [13] of the host when the suspicious behaviors are detected.

3.2 Cyber Attack Prevention Description

As shown in Fig. 4, when a suspicious insider conducts suspicious behaviors on the network, such as logging in to a protected system where he or she has no

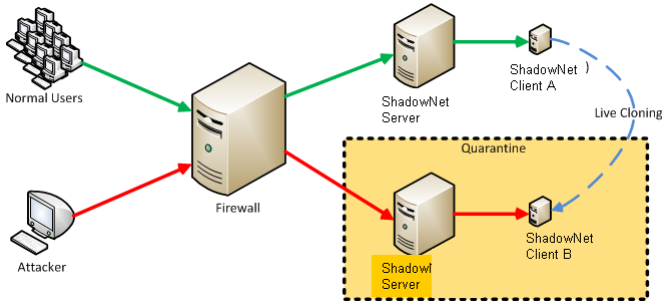


Fig. 2. ShadowNet Active Defense Network Topology

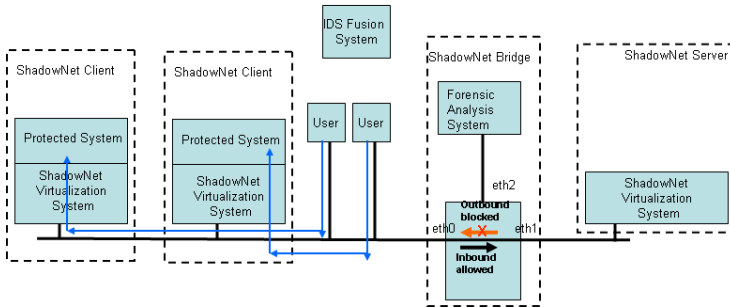


Fig. 3. ShadowNet Active Defense Infrastructure

legitimate access rights or logging into the system at an abnormal time, his behaviors will trigger the intrusion detection sensors installed on the network, which in turn sends out alarm messages to the IDS fusion system. The fusion system sends control messages to the "ShadowNet Client" and "Shadow Server" for system live clone action. As shown in Fig. 5, a live clone [11] of the system that is breached by the suspicious insider will be prepared in real-time by the ShadowNet Client. This clone system will have exactly same status, file system structure, and network availability as the original system, but will not contain the original host's sensitive data. All these sensitive data are eliminated and replaced with fraud data that contain useless information. The system clone along with the suspicious network connection will be migrated to another physical server (ShadowNet Server) without noticed by the attacker. Using the live system migration technology, the total time for migrating the attacker to the Server will be within 100 ms. As shown in Fig. 5, after the migration, the ShadowNet Client will automatically re-route all network connection package from the suspicious insider to the migrated system clone in ShadowNet Server behind the "ShadowNet Bridge".

The migrated clone has same environment of the original host that the insider breached. Thus, the suspicious user or insider will not aware that his/her

connection has been migrated to a closely monitored forensic analysis platform. A ShadowNet Bridge can be deployed to quarantine the ShadowNet server by preventing the potential of information leak out. The ShadowNet Bridge is special designed ISO second level system that are transparent in the network and allow inbound network connection in but stop all outbound connection initial from behind ShadowNet Bridge. This prevents the protected system clone from being used by the attacker as an ad-hoc attacking machine for attacking other machine. All network transactions and communications to the system clone can be collected by ShadowNet bridge for future forensic analysis.

3.3 Core Technologies

The innovative technology behind the ShadowNet system is a real-time system live clone and migration technology called the ShadowNet live clone. The technology enables the ShadowNet system construct a system clone, and to move the system clone along with the suspicious network connection to another physical server (ShadowNet Server) without being noticed by the attacker. At the same time, the original system and other users of the system are not impacted.

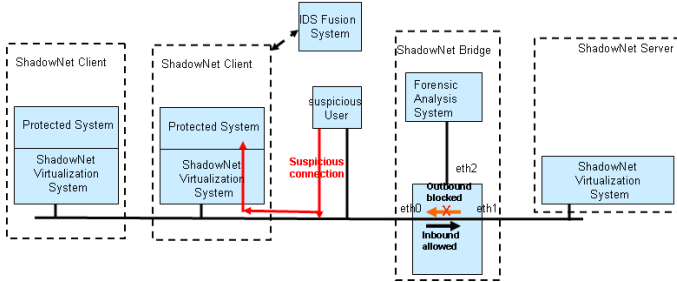


Fig. 4. Suspicious Connections from Insider User’s Computer

Different from the traditional visualization technology [12], our system needs the capability to both migrate operating system instances across distinct physical hosts, and also to keep the current operating system continually running to support the normal business. This requires the change of MAC address and IP address of the live virtual clone. Most of the original system network connections are maintained and only the suspicious network connection ports are disabled or disconnected. By modifying Xen Virtual machine platform, we built an experiment implementation to demonstrate the ShadowNet system. As shown in Fig. 6. In the implementation, we use a Virtual Machine (VM) Descriptor as a condensed VMimage that allows swift VM replication to a separate physical host. Construction of a VM descriptor starts by spawning a thread in the VM kernel issues a hypercall suspending the VM’s execution. When the hypercall succeeds, a privileged process in domain0 maps the suspended VM memory to populate

the descriptor. The descriptor contains: (1) metadata describing the VM and its virtual devices, (2) a few memory pages shared between the VM and the VM hypervisor, (3) the registers of the main VCPU, (4) the Global Descriptor Tables (GDT) used by the x86 segmentation hardware for memory protection, and (5) the page tables of the VM. A monitor and control tool, SXMaster, is developed and is capable of receiving the alert from the existing IDS enclave to trigger the clone migration process that moves the suspicious user’s system and network connection to quarantined ShadowNet server. SXMaster uses socket-based communications to execute commands on the ShadowNet infrastructure machines. Each instance of the application listens by default on port 4445 for incoming connections from other instances of the application. The software uses iptables commands to configure NAT, and uses contrack-tools commands to assist in live session migration.

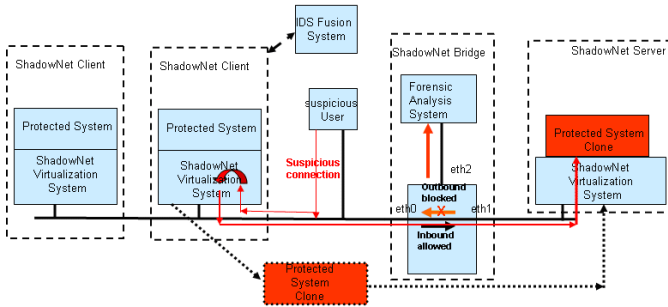


Fig. 5. Active defense for Attack Prevention and Forensics Collections

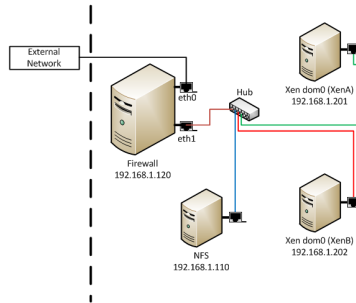


Fig. 6. Experiment Network Topology

4 Conclusion

According to our research, there are currently no COTS systems that can provide effective mechanisms to prevent insider attack. Most of the work associated with

preventing the insider attack focuses on studying security policies and policy enforcement. The ShadowNet technology provides a system that will be able to prevent the attack from suspicious insider who has knowledge about the network structure and the location of the sensitive information. The ShadowNet system also provides a real-time forensics data gathering capability to support large, geographically dispersed networks without disturbing the system's operations. This capability will aid in real-time attack analysis, countermeasures development, and legal prosecution.

Existing information security technologies such as firewalls or Intrusion Detection Systems (IDS) cannot provide an adequate defense against insider threats because they are oriented towards attacks originated from outside the enterprise. Insider attacks may begin from any of numerous potential attack points in the enterprise and have too many parameters to be monitored that existing systems cannot handle. By implementing the ShadowNet infrastructure developed in this research, the whole network becomes a distributed IDS grid. Any machine in the network is a potential honeypot to quarantine a malicious insider.

Acknowledgment. This work was supported in part by the Oak Ridge National Laboratory LDRD program. The views and conclusions contained in this document are those of the authors. This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

References

1. Salem, M.B., Hershkop, S., Stolfo, S.J.: A Survey of Insider Attack Detection Research. *Advances in Information Security* 39, 69–90 (2008)
2. The eighth annual CSI/FBI 2003 report: Computer Crime and Security Survey (2003)
3. Stone, C.: Information Sharing in the Era of WikiLeaks: Balancing Security and Collaboration, Office of The Director of National Intelligence, Washington, DC (March 2011)
4. Bellovin, S.: The Insider Attack Problem Nature and Scope. *Advances in Information Security* 39, 69–90 (2008)
5. Braz, F.A., Fernandez, E.B., VanHilst, M.: Eliciting Security Requirements through Misuse Activities. In: *Proceedings of the 2008 19th International Conference on Database and Expert Systems Application (DEXA)*, pp. 328–333 (2008)
6. Bellovin, S.: There Be Dragons. In: *Proc. of the Third Usenix Security Symposium*, Baltimore MD (September 1992)
7. Bellovin, S.M.: Packets Found on an Internet. *Computer Communications Review* 23(3), 26–31 (July)
8. Spitzner, L.: Honeypots: Catching the Insider Threat. In: *19th Annual Computer Security Applications Conference (ACSAC 2003)*, p. 170 (2003)

9. Spitzner, L.: Honeypots: Tracking Hackers. Addison-Wesley Longman Publishing Co., Inc., Boston (2002)
10. Lyon, G.: Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security. Insecure Publisher, USA (2009) ISBN 9780979958717
11. Sun, Y., Luo, Y., Wang, X., Wang, Z., Zhang, B., Chen, H., Li, X.: Fast Live Cloning of Virtual Machine Based on Xen. In: 2009 11th IEEE International Conference on High Performance Computing and Communications, HPCC 2009, pp. 392–399 (2009)
12. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: Proceedings of the ACM Symposium on Operating Systems Principles (October 2003)
13. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI), Boston, MA, pp. 273–286 (May 2005)

Author Index

- Abdullah, Noryusliza IV-364
Abdullah, Nurul Azma IV-353
Abid, Hassan III-368
Adewumi, Adewole IV-248
Afonso, Vitor Monte IV-274, IV-302
Agarwal, Suneeta IV-147
Aguiar, Rui L. III-682
Aguilar, José Alfonso IV-116
Ahmadian, Kushan I-188
Alarcon, Vladimir J. II-578, II-589
Albuquerque, Caroline Oliveira III-576
Alfaro, Pablo IV-530
Ali, Salman III-352
Alizadeh, Hosein III-647
Almeida, Regina III-17
Alonso, Pedro I-29
Alves, Daniel S.F. I-101
Amandi, Analía A. III-698, III-730
Amaral, Paula III-159
Amjad, Jaweria III-368
Ammar, Reda A. I-161
Amorim, Elisa P. dos Santos I-635
An, Deukhyeon III-272
Anderson, Roger W. I-723
Angeloni, Marcus A. I-240
Antonino, Pablo Oliveira III-576
Aquilanti, Vincenzo I-723
Arefin, Ahmed Shamsul I-71
Armentano, Marcelo G. III-730
Arnaout, Arghad IV-392
Aromando, Angelo III-481
Asche, Hartmut II-347, II-386, II-414, II-439
Aydin, Ali Orhan IV-186
Ayres, Rodrigo Moura Juvenil III-667
Azad, Md. Abul Kalam III-72
Azzato, Antonello II-686
- Bae, Sunwook III-238
Balena, Pasquale I-583, II-116
Balucani, Nadia I-331
Barbosa, Ciro I-707
Barbosa, Fernando Pires IV-404
Barbosa, Helio J.C. I-125
- Baresi, Umberto II-331
Barreto, Marcos I-29
Baruque, Alexandre Or Cansian IV-302
Bastianini, Riccardo I-358
Batista, Augusto Herrmann III-631
Batista, Vitor A. IV-51
Battino, Silvia II-624
Beaver, Justin IV-646
Bencardino, Massimiliano II-548
Berdún, Luis III-698
Berenguel, José L. III-119
Bernardino, Heder S. I-125
Berretta, Regina I-71
Biehl, Matthias IV-40
Bimonte, Sandro II-373
Bisceglie, Roberto II-331
Bitencourt, Ana Carla P. I-723
Blecic, Ivan II-481, II-492
Boavida, Fernando II-234
Bollini, Letizia II-508
Boratto, Murilo I-29
Borg, Erik II-347, III-457
Borruso, Giuseppe II-624, II-670
Boulil, Kamal II-373
Braga, Ana Cristina I-665
Brumana, Raffaella II-397
Bugs, Geisa I-477
Burgarelli, Denise I-649
Bustos, Víctor III-607
- Caiaffa, Emanuela II-532
Calazan, Rogério M. I-148
Caldas, Daniel Mendes I-675
Callejo, Miguel-Ángel Manso I-462
Camarda, Domenico II-425
Campobasso, Francesco II-71
Campos, Ricardo Silva I-635
Candori, Pietro I-316, I-432
Cannatella, Daniele II-54
Cano, Marcos Daniel III-743
Cansian, Adriano Mauro IV-286
Carbonara, Sebastiano II-128
Cardoso, João M.P. IV-217
Carmo, Rafael IV-444

- Carneiro, Joubert C. Lima e Tiago G.S. II-302
- Carpené, Michele I-345
- Carvalho, Luis Paulo da Silva II-181
- Carvalho, Maria Sameiro III-30, III-187
- Casado, Leocadio G. III-119, III-159
- Casado, Leocadio Gonzalez I-57
- Casas, Giuseppe B. Las II-466, II-640, II-686
- Casavecchia, Piergiorgio I-331
- Castro, Patrícia F. IV-379
- Cavalcante, Gabriel D. IV-314
- Cecchi, Marco I-345
- Cecchini, Arnaldo II-481, II-492
- Ceppi, Claudia II-517
- Cermignani, Matteo I-267
- Cerreta, Maria II-54, II-168, II-653
- Chanet, Jean-Pierre II-373
- Charão, Andrea Schwertner IV-404
- Cho, Yongyun IV-613, IV-622
- Cho, Young-Hwa IV-543
- Choe, Junseong III-324
- Choi, Hong Jun IV-602
- Choi, Jae-Young IV-543
- Choi, Jongsun IV-613
- Choi, Joonsoo I-214
- Choo, Hyunseung III-259, III-283, III-324
- Chung, Tai-Myoung III-376
- Ci, Song III-297
- Cicerone, Serafino I-267
- Ciloglugil, Birol III-550
- Cioquetta, Daniel Souza IV-16
- Clarke, W.A. IV-157
- Coelho, Leandro I-29
- Coletti, Cecilia I-738
- Conrado, Merley da Silva III-618
- Corea, Federico-Vladimir Gutiérrez I-462
- Coscia, José Luis Ordiales IV-29
- Costa, M. Fernanda P. III-57, III-103
- Costantini, Alessandro I-345, I-401, I-417
- Crasso, Marco IV-29, IV-234, IV-484
- Crawford, Broderick III-607
- Crocchianti, Stefano I-417
- Cuca, Branka II-397
- Cui, Xiaohui IV-646
- Cunha, Jácome IV-202
- da Luz Rodrigues, Francisco Carlos III-657
- Danese, Maria III-512
- Dantas, Sócrates de Oliveira I-228
- da Silva, Paulo Caetano II-181
- Daskalakis, Vangelis I-304
- de Almeida, Ricardo Aparecido Perez IV-470, IV-560
- de Avila, Ana Maria H. III-743
- de By, Rolf A. II-286
- de Carvalho, Andre Carlos P.L.F. III-562
- de Carvalho Jr., Osmar Abílio III-657
- Decker, Hendrik IV-170
- de Costa, Evandro Barros III-714
- de Deus, Raquel Faria II-565
- de Felice, Annunziata II-1
- de Geus, Paulo Lício IV-274, IV-302, IV-314
- Delgado del Hoyo, Francisco Javier I-529
- Dell'Orco, Mauro II-44
- de Macedo Mourelle, Luiza I-101, I-113, I-136, I-148
- de Magalhães, Jonathas José III-714
- De Mare, Gianluigi II-27
- Dembogurski, Renan I-228
- de Mendonça, Rafael Mathias I-136
- de Miranda, Péricles B.C. III-562
- de Oliveira, Isabela Liane IV-286
- de Oliveira, Wellington Moreira I-561
- de Paiva Oliveira, Alcione I-561
- Deris, Mustafa Mat I-87, IV-340
- De Santis, Fortunato III-481
- Désidéri, Jean-Antoine IV-418
- de Souza, Cleyton Caetano III-714
- de Souza, Éder Martins III-657
- de Souza, Renato Cesar Ferreira I-502
- de Souza Filho, José Luiz Ribeiro I-228, II-712
- De Toro, Pasquale II-168
- Dias, Joana M. III-1
- Dias, Luis III-133
- do Nascimento, Gleison S. IV-67
- Donato, Carlo II-624
- do Prado, Hércules Antonio III-631, III-657
- dos Anjos, Eudisley Gomes IV-132
- dos Santos, Jefersson Alex I-620

- dos Santos, Rafael Duarte Coelho IV-274, IV-302
- dos Santos, Rodrigo Weber I-635, I-649, I-691, I-707
- dos Santos Soares, Michel IV-1, IV-16
- e Alvelos, Filipe Pereira III-30
- El-Attar, Mohamed IV-258
- Elish, Mahmoud O. IV-258
- El-Zawawy, Mohamed A. III-592, IV-83
- Engemaier, Rita II-414
- Eom, Young Ik III-227, III-238, III-272
- Epicoco, Italo I-44
- Esmael, Bilal IV-392
- Ezzatti, Pablo IV-530
- Falcinelli, Stefano I-316, I-331, I-387, I-432
- Falcone, Roberto II-508
- Fanizzi, Annarita II-71
- Farage, Michèle Cristina Resende I-675
- Farantos, C. Stavros I-304
- Farias, Matheus IV-444
- Fechine, Joseana Macêdo III-714
- Fedel, Gabriel de S. I-620
- Felino, António I-665
- Fernandes, Edite M.G.P. III-57, III-72, III-103
- Fernandes, Florbela P. III-103
- Fernandes, João P. IV-202, IV-217
- Ferneda, Edilson III-631, III-657
- Ferreira, Ana C.M. III-147
- Ferreira, Brigida C. III-1
- Ferreira, Manuel III-174
- Ferroni, Michele I-358
- Fichtelmann, Bernd II-347, III-457
- Fidêncio, Érika II-302
- Figueiredo, José III-133
- Filho, Dario Simões Fernandes IV-274, IV-302
- Filho, Jugurta Lisboa I-561
- Fiorese, Adriano II-234
- Fonseca, Leonardo G. I-125
- Formosa, Saviour II-609
- Formosa Pace, Janice II-609
- Fort, Marta I-253
- França, Felipe M.G. I-101
- Freitas, Douglas O. IV-470
- Fruhwrith, Rudolf K. IV-392
- García, I. III-119
- García, Immaculada I-57
- Garrigós, Irene IV-116
- Gasior, Wade IV-646
- Gavrilova, Marina I-188
- Gentili, Eleonora III-539
- Geraldes, Carla A.S. III-187
- Gervasi, Osvaldo IV-457
- Ghandehari, Mehran II-194
- Ghazali, Rozaida I-87
- Ghiselli, Antonia I-345
- Ghizoni, Maria Luísa Amarante IV-588
- Girard, Luigi Fusco II-157
- Gomes, Ruan Delgado IV-132
- Gomes, Tiago Costa III-30
- Gonschorek, Julia II-208, II-220
- Gonzaga de Oliveira, Sanderson Lincohn I-172, I-198, I-610
- Görlich, Markus I-15
- Greco, Ilaria II-548
- Grégio, André Ricardo Abed IV-274, IV-286, IV-302
- Guardia, Hélio C. IV-560
- Gupta, Pankaj III-87
- Hahn, Kwang-Soo I-214
- Haijema, Rene III-45
- Han, Jikwang III-217
- Han, JungHyun III-272
- Han, Yanni III-297
- Handaga, Bana IV-340
- Hasan, Osman III-419
- Hashim, Rathiah II-728
- Hendrix, Eligius M.T. I-57, III-45, III-119, III-159
- Heo, Jaewee I-214
- Hong, Junguye III-324
- Huang, Lucheng I-447
- Ibrahim, Rosziati IV-353, IV-364
- Igounet, Pablo IV-530
- Ikhu-Omoregbe, Nicholas IV-248
- Im, Illkyun IV-543
- Imtiaz, Sahar III-339
- Inceoglu, Mustafa Murat III-550
- Iochpe, Cirano IV-67
- Ipbuker, Cengizhan III-471
- Ivánová, Ivana II-286
- Izkara, Jose Luis I-529

- Jeon, Jae Wook III-311
 Jeon, Woongryul III-391
 Jeong, Jongpil IV-543
 Jeong, Soonmook III-311
 Jino, Mario IV-274, IV-302
 Jorge, Eduardo IV-444
 Jung, Sung-Min III-376
- Kalsing, André C. IV-67
 Kang, Min-Jae III-217
 Karimipour, Farid II-194
 Kasprzak, Andrzej IV-514, IV-576
 Kaya, Sinasi III-471
 Khalid, Noor Elaiza Abdul II-728
 Khan, Salman H. III-339
 Khan, Yasser A. IV-258
 Khanh Ha, Nguyen Phan III-324
 Kim, Cheol Hong IV-602
 Kim, Hakhyun III-391
 Kim, Iksu IV-622
 Kim, Jeehong III-227, III-238, III-272
 Kim, Junho I-214
 Kim, Young-Hyuk III-248
 Kischinhevsky, Mauricio I-610
 Kluge, Mario II-386
 Knop, Igor I-707
 Komati, Karin S. II-739
 Kopeliovich, Sergey I-280
 Kosowski, Michał IV-514
 Koszalka, Leszek IV-576
 Koyuncu, Murat IV-234
 Kwak, Ho-Young III-217
 Kwon, Keyho III-311
 Kwon, Ki-Ryong IV-434
 Kwon, Seong-Geun IV-434
- Ladeira, Pitter Reis II-548
 Laganà, Antonio I-292, I-345, I-358,
 I-371, I-387, I-401, I-417
 Lago, Noelia Faginas I-387
 Laguna Gutiérrez, Víctor Antonio
 III-618
 Lanorte, Antonio III-481, III-512
 Lanza, Viviana II-686
 Lasaponara, Rosa III-481, III-497,
 III-512
 Le, Duc Tai III-259
 Lederer, Daniel II-263
 Le Duc, Thang III-259
 Lee, Dong-Young III-368, III-376
- Lee, Eung-Joo IV-434
 Lee, Hsien-Hsin IV-602
 Lee, Jae-Gwang III-248
 Lee, Jae-Kwang III-248
 Lee, Jae-Pil III-248
 Lee, Jongchan IV-613
 Lee, Junghoon III-217
 Lee, Kwangwoo III-391
 Lee, Sang Joon III-217
 Lee, Suk-Hwan IV-434
 Lee, Yunho III-391
 Leonel, Gildo de Almeida II-712
 Leonori, Francesca I-331
 Li, Yang III-297
 Lim, Il-Kwon III-248
 Lima, Priscila M.V. I-101
 Lin, Tao III-297
 Liu, Yi IV-100
 Lobarinhas, Pedro III-202
 Lobosco, Marcelo I-675, I-691, I-707
 Loconte, Pierangela II-517
 Lomba, Ricardo III-202
 Lombardi, Andrea I-387
 Lopes, Maria do Carmo III-1
 Lopes, Paulo IV-217
 Lou, Yan I-447
 Lubisco, Giorgia II-517
 Luiz, Alfredo José Barreto III-657
- Ma, Zhiyi IV-100
 Macedo, Gilson C. I-691, I-707
 Maffioletti, Sergio I-401
 Maleki, Behzad III-647
 Mancini, Francesco II-517
 Mangialardi, Giovanna II-116
 Manuali, Carlo I-345
 Marcondes, Cesar A.C. IV-470
 Marghany, Maged III-435, III-447
 Marimbaldo, Francisco-Javier Moreno
 I-462
 Marinho, Euler Horta IV-632
 Martins, Luís B. III-147
 Martins, Pedro IV-217
 Martucci, Isabella II-1
 Marucci, Alessandro II-532
 Marwala, T. IV-157
 Marwedel, Peter I-15
 Marzuoli, Annalisa I-723
 Mashkoor, Atif III-419
 Masini, Nicola III-497

- Mateos, Cristian IV-29, IV-234, IV-484
 Mazhar, Aliya III-368
 Mazón, Jose-Norberto IV-116
 McAnally, William H. II-578, II-589
 Medeiros, Claudia Bauzer I-620
 Mehlawat, Mukesh Kumar III-87
 Meira Jr., Wagner I-649
 Mele, Roberta II-653
 Melo, Tarick II-302
 Messine, F. III-119
 Miao, Hong I-447
 Milani, Alfredo III-528, III-539
 Min, Changwoo III-227, III-238
 Min, Jae-Won III-376
 Misra, A.K. IV-157
 Misra, Sanjay IV-29, IV-147, IV-234,
 IV-248
 Miziolek, Marek IV-514
 Mocavero, Silvia I-44
 Mohamad, Kamaruddin Malik IV-353
 Mohamed Elsayed, Samir A. I-161
 Monfroy, Eric III-607
 Monteserin, Ariel III-698
 Montrone, Silvestro II-102
 Moreira, Adriano II-450
 Moreira, Álvaro IV-67
 Moscato, Pablo I-71
 Moschetto, Danilo A. IV-470
 Müller, Heinrich I-15
 Mundim, Kleber Carlos I-432
 Mundim, Maria Suelly Pedrosa I-316
 Mundim, Maria Suely Pedrosa I-432
 Munir, Ali III-352, III-368
 Murgante, Beniamino II-640, II-670,
 III-512
 Murri, Riccardo I-401
 Musaoglu, Nebiye III-471

 Nabwey, Hossam A. II-316, II-358
 Nakagawa, Elisa Yumi III-576
 Nalli, Danilo I-292
 Nawi, Nazri Mohd I-87
 Nedjah, Nadia I-101, I-113, I-136, I-148
 Nema, Jigyasu III-528
 Neri, Igor IV-457
 Nesticò, Antonio II-27
 Neves, Brayan II-302
 Nguyên, Toàn IV-418
 Niu, Wenjia III-297
 Niyogi, Rajdeep III-528

 Nolè, Gabriele III-512
 Nunes, Manuel L. III-147

 O'Kelly, Morton E. II-249
 Oliveira, José A. III-133
 Oliveira, Rafael S. I-649
 Oreni, Daniela II-397
 Ottomanelli, Michele II-44

 Pacifici, Leonardo I-292, I-371
 Pádua, Clarindo Isaías P.S. IV-51
 Pádua, Wilson IV-51
 Pallottelli, Simonetta I-358
 Panaro, Simona II-54
 Pandey, Kusum Lata IV-147
 Paolillo, Pier Luigi II-331
 Park, Changyong III-283
 Park, Gyung-Leen III-217
 Park, Junbeom III-283
 Park, Sangjoon IV-613
 Park, Young Jin IV-602
 Parvin, Hamid III-647
 Parvin, Sajad III-647
 Pathak, Surendra II-589
 Pauls-Worm, Karin G.J. III-45
 Peixoto, Daniela C.C. IV-51
 Peixoto, João II-450
 Pepe, Monica II-397
 Perchinunno, Paola II-88, II-102
 Pereira, Gilberto Corso I-491
 Pereira, Guilherme A.B. III-133, III-187
 Pereira, Óscar Mortágua III-682
 Pereira, Tiago F. I-240
 Pessanha, Fábio Gonçalves I-113
 Pigozzo, Alexandre B. I-691, I-707
 Pimentel, Dulce II-565
 Pingali, Keshav I-1
 Pinheiro, Marcello Sandi III-631
 Pirani, Fernando I-316, I-387, I-432
 Piscitelli, Claudia II-517
 Poggioni, Valentina III-539
 Pol, Maciej IV-576
 Pollino, Maurizio II-532
 Poma, Lourdes P.P. IV-470
 Pontrandolfi, Piergiuseppe II-686
 Poplin, Alenka I-491
 Pozniak-Koszalka, Iwona IV-576
 Pradel, Marilys II-373
 Prasad, Rajesh IV-147
 Prieto, Iñaki I-529

- Proma, Wojciech IV-576
 Prudêncio, Ricardo B.C. III-562
- Qadir, Junaid III-352
 Qaisar, Saad Bin III-339, III-352,
 III-407
 Quintela, Bárbara de Melo I-675, I-691,
 I-707
- Ragni, Mirco I-723
 Raja, Haroon III-368, III-407
 Rajasekaran, Sanguthevar I-161
 Rak, Jacek IV-498
 Ramiro, Carla I-29
 Rampini, Anna II-397
 Rasekh, Abolfazl II-275
 Re, Nazzareno I-738
 Renhe, Marcelo Caniato II-712
 Resende, Rodolfo Ferreira IV-632
 Rezende, José Francisco V. II-302
 Rezende, Solange Oliveira III-618
 Ribeiro, Hugo IV-202
 Ribeiro, Marcela Xavier III-667, III-743
 Riveros, Carlos I-71
 Rocha, Ana Maria A.C. III-57, III-72,
 III-147
 Rocha, Bernardo M. I-649
 Rocha, Humberto III-1
 Rocha, Jorge Gustavo I-571
 Rocha, Maria Célia Furtado I-491
 Rocha, Pedro Augusto F. I-691
 Rodrigues, António M. II-565
 Romani, Luciana A.S. III-743
 Rosi, Marzio I-316, I-331
 Rossi, Elda I-345
 Rossi, Roberto III-45
 Rotondo, Francesco I-545
 Ruiz, Linnyer Beatrys IV-588
- Sad, Dhiego Oliveira I-228
 Salles, Evandro O.T. II-739
 Salles, Ronaldo M. IV-326
 Salvatierra, Gonzalo IV-484
 Sampaio-Fernandes, João C. I-665
 Samsudin, Nurnabilah II-728
 Sanches, Silvio Ricardo Rodrigues
 II-699
 Sanjuan-Estrada, Juan Francisco I-57
 Santos, Aduino IV-588
 Santos, Maribel Yasmina III-682
 Santos, Marilde Terezinha Prado
 III-667, III-743
 Santos, Teresa II-565
 Sarafian, Haiduke I-599
 Saraiva, João IV-202, IV-217
 Sarcinelli-Filho, Mario II-739
 Schirone, Dario Antonio II-1, II-17,
 II-88
 Sciberras, Elaine II-609
 Scorza, Francesco II-640
 Selicato, Francesco I-545, II-517
 Selicato, Marco II-144
 Sellarès, J. Antoni I-253
 Sertel, Elif III-471
 Shaaban, Shaaban M. II-316, II-358
 Shah, Habib I-87
 Shukla, Mukul IV-157
 Shukla, Ruchi IV-157
 Silva, António I-571
 Silva, João Tácio C. II-302
 Silva, José Eduardo C. I-240
 Silva, Rodrigo I-228
 Silva, Rodrigo M.P. IV-326
 Silva, Roger Correia I-228
 Silva, Valdinei Freire da II-699
 Silva Jr., Luneque I-113
 Silvestre, Eduardo Augusto IV-1
 Simões, Flávio O. I-240
 Simões, Paulo II-234
 Singh, Gaurav II-286
 Skouteris, Dimitris I-331
 Soares, Carlos III-562
 Song, Hokwon III-227, III-238
 Song, Taehoun III-311
 Soravia, Marco II-599
 Soto, Ricardo III-607
 Souza, Cleber P. IV-314
 Stanganelli, Marialuce II-599
 Stankute, Silvija II-439
- Tajani, Francesco II-27
 Tasso, Sergio I-358, IV-457
 Tavares, Maria Purificação I-665
 Teixeira, Ana Paula III-17
 Teixeira, José Carlos III-174, III-202
 Teixeira, Senhorinha F.C.F. III-147,
 III-202
 Tentella, Giorgio I-417
 Thom, Lucinéia IV-67

- Thonhauser, Gerhard IV-392
 Tilio, Lucia II-466, II-686
 Timm, Constantin I-15
 Tori, Romero II-699
 Torkan, Germano II-17
 Torre, Carmelo Maria I-583, II-116,
 II-144, II-157
 Traina, Agma J.M. III-743
 Treadwell, Jim IV-646
 Tricaud, Sebastien IV-314
 Trofa, Giovanni La II-144
 Trunfio, Giuseppe A. II-481, II-492
 Tsoukiàs, Alexis II-466
 Tyrallová, Lucia II-208, II-220

 Usera, Gabriel IV-530

 Vafaeinezhad, Ali Reza II-275
 Vargas-Hernández, José G. I-518
 Varotsis, Constantinos I-304
 Vaz, Paula I-665
 Vecchiocattivi, Franco I-316, I-432
 Vella, Flavio IV-457
 Verdicchio, Marco I-371
 Verma, Shilpi III-87
 Versaci, Francesco I-1
 Vieira, Marcelo Bernardes I-228, II-712

 Vieira, Wesley IV-444
 Vyatkina, Kira I-280

 Walkowiak, Krzysztof IV-498, IV-514
 Wang, Hao III-283
 Wang, Kangkang I-447
 Weichert, Frank I-15
 Won, Dongho III-391
 Wu, Feifei I-447

 Xavier, Carolina R. I-635
 Xavier, Micael P. I-691
 Xexéo, Geraldo B. IV-379
 Xu, Yanmei I-447
 Xu, Yuemei III-297

 Yanalak, Mustafa III-471

 Zaldívar, Anibal IV-116
 Zenha-Rela, Mário IV-132
 Zhang, Tian IV-100
 Zhang, Xiaokun IV-100
 Zhang, Yan IV-100
 Zhao, Xuying IV-100
 Zito, Romina I-583
 Zunino, Alejandro IV-29, IV-234,
 IV-484