

Services Science

LNCS 9435

Alistair Barros
Daniela Grigori
Nanjangud C. Narendra
Hoa Khanh Dam (Eds.)

Service-Oriented Computing

13th International Conference, ICSOC 2015
Goa, India, November 16–19, 2015
Proceedings



Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, Lancaster, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Zürich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbrücken, Germany

More information about this series at <http://www.springer.com/series/7408>

Alistair Barros · Daniela Grigori
Nanjangud C. Narendra · Hoa Khanh Dam (Eds.)

Service-Oriented Computing

13th International Conference, ICSOC 2015
Goa, India, November 16–19, 2015
Proceedings

Editors

Alistair Barros
Queensland University of Technology
Brisbane, QLD
Australia

Daniela Grigori
Université Paris Dauphine
Paris
France

Nanjangud C. Narendra
M.S. Ramaiah University
Bangalore
India

Hoa Khanh Dam
University of Wollongong
Wollongong, NSW
Australia

ISSN 0302-9743

ISSN 1611-3349 (electronic)

Lecture Notes in Computer Science

ISBN 978-3-662-48615-3

ISBN 978-3-662-48616-0 (eBook)

DOI 10.1007/978-3-662-48616-0

Library of Congress Control Number: 2015947417

LNCS Sublibrary: SL2 – Programming and Software Engineering

Springer Heidelberg New York Dordrecht London

© Springer-Verlag Berlin Heidelberg 2015

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, 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.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer-Verlag GmbH Berlin Heidelberg is part of Springer Science+Business Media
(www.springer.com)

Preface

Welcome to the proceedings of the 13th International Conference on Service-Oriented Computing (ICSOC 2015), held in Goa, India, November 16–19, 2015. ICSOC 2015 was co-organized by ServTech and IBM Research. These proceedings contain high-quality research papers that represent the latest results, ideas, and position and vision statements in the field of service-oriented computing.

Since the first occurrence of the conference in 2003, where service-oriented computing was given impetus through the rise of Web service platforms and standards as well as dedicated service description and composition languages and techniques, ICSOC has grown to become a premier international forum. It has continued to attract high-quality and highly cited publications from academics, industry researchers, and practitioners to share, report, and discuss their ground-breaking work. ICSOC 2015 continued along this tradition.

The selected papers demonstrate increasing maturity and synergies across service-oriented computing and cloud computing, growing synergies with the Internet of Things, and ongoing maturity in service composition and interoperability, which are strongly related to the fields of BPM and software engineering. This year's call for papers attracted 124 research and industry submissions from 31 countries and six continents. The submissions were rigorously evaluated by at least three reviewers, followed by a discussion moderated by a senior Program Committee (PC) member who made a final recommendation in the form of a meta-review. The PC was composed of 148 world-class experts (136 PC members and 22 senior PC members) in service-oriented computing from 22 different countries.

The ICSOC 2015 program featured 18 full papers (acceptance rate of 16 %) and nine short papers in the research track. Also featured were five papers in the industry track. The conference program was highlighted by invited keynotes, lively panel discussions, multiple demonstrations, the PhD Symposium, and eight workshops on different aspects of service-oriented and cloud computing.

We would like to express our gratitude to all individuals, institutions, and sponsors that supported ICSOC 2015. The high-quality program would not have been possible without the expertise and dedication of our PC and in particular of our senior PC members. We are grateful for the guidance of the general chairs (Aditya Ghose and Srinivas Padmanabhuni), the effort of more than 60 external reviewers, the proceedings chair (Hoa Dam), the publicity chairs (Georgiana Copil, Tri Kurniawan and Renuka Sindhgatta), the local organizers (Karthikeyan Ponnalagu and Wagh Ramarao S.) and volunteers, and last but not least to the distinguished members of the ICSOC Steering Committee. All of them helped to make ICSOC 2015 a success. Finally, a special word of thanks goes to all researchers, practitioners, and students who contributed their presentations, questions, and active participation in the conference. We hope you enjoy these proceedings!

November 2015

Alistair Barros
Daniela Grigori
Nanjangud C. Narendra

ICSOC 2015 Organization

General Chairs

Aditya Ghose	University of Wollongong, Australia
Srinivas Padmanabhuni	Infosys Technologies Limited, India

Program Chairs

Alistair Barros	Queensland University of Technology, Australia
Daniela Grigori	Université Paris Dauphine, France
Nanjangud C. Narendra	M.S. Ramaiah University, India

Steering Committee

Boualem Benatallah	University of New South Wales, Australia
Fabio Casati	University of Trento, Italy
Bernd Krämer	FernUniversität in Hagen, Germany
Winfried Lamersdorf	University of Hamburg, Germany
Heiko Ludwig	IBM Research, USA
Mike Papazoglou	Tilburg University, The Netherlands
Jian Yang	Macquarie University, Australia
Liang Zhang	Fudan University, China

Publication Chair

Hoa Khanh Dam	University of Wollongong, Australia
---------------	-------------------------------------

Workshop Chairs

Walid Gaaloul	Telecom SudParis, France
Alex Norton	Tallinn University of Technology, Estonia
G.R. Gangadharan	IDRBT, India

Panel Chairs

Florian Daniel	University of Trento, Italy
Gargi B. Dasgupta	IBM Research, India
Andreas Metzger	University of Duisburg-Essen, Germany

Finance Chair

Bernd Krämer FernUniversität in Hagen, Germany

PhD Symposium Chairs

Antonio Brogi University of Pisa, Italy
Hong-Linh Truong TU Wien, Austria
Guido Governatori NICTA, Australia

Demonstration Track Chairs

Vinay Kulkarni Tata Consultancy Services Limited, India
P. Radha Krishna Infosys Limited, India
Vinod Muthusamy IBM Research, USA

Organizing Chairs

Karthikeyan Ponnalagu IBM Research, India
Wagh Ramrao S. Goa University, India

Publicity Chairs

Georgiana Copil TU Wien, Austria
Tri Kurniawan Brawijaya University, Indonesia
Renuka Sindhgatta IBM Research, India

Web Chairs

Allahbaksh Asadullah Infosys Labs, India
Ayu Saraswati University of Wollongong, Australia

Senior Program Committee

Samik Basu Iowa State University, USA
Boualem Benatallah UNSW, Australia
Athman Bouguettaya RMIT, Australia
Fabio Casati University of Trento, Italy
Schahram Dustdar TU Wien, Austria
Xavier Franch Universitat Politecnica de Catalunya, Spain
Aditya Ghose University of Wollongong, Australia
Mohand-Said Hacid University of Lyon, France
Grace Lewis Software Engineering Institute, USA
Lin Liu Tsinghua University, China
Heiko Ludwig IBM Research, USA
Michael Maximilien IBM Research, USA

Flavio De Paoli	Università di Milano Bicocca, Italy
Cesare Pautasso	University of Lugano, Switzerland
Barbara Pernici	Politecnico di Milano, Italy
Gustavo Rossi	UNLP, Argentina
Michael Q. Sheng	Adelaide University, Australia
Stefan Tai	TU Berlin, Germany
Zahir Tari	RMIT University, Australia
Mathias Weske	HPI/University of Potsdam, Germany
Jian Yang	Macquarie University, Australia
Liang Zhang	Fudan University, China

Program Committee

Fahim Akhter	King Saud University, Saudi Arabia
Rama Akkiraju	IBM/USA, USA
Alvaro Arenas	Instituto de Empresa Business School, Spain
Ebrahim Bagheri	Ryerson University, Canada
Luciano Baresi	Politecnico di Milano, Italy
Alistair Patrick Barros	Queensland University of Technology, Australia
N. Md. Jubair Basha	Muffakham Jah College of Engineering and Technology, India
Djamel Belaid	Telecom Sud Paris, France
Khalid Belhajjame	Université Paris Dauphine, France
Umesh Bellur	Indian Institute of Technology, India
Nejib Ben Hadj-Alouane	National School of Engineers of Tunis (ENIT), Tunisia
Moez Ben Haj Hmida	National Engineering School of Tunis (ENIT), Tunisia
Sonia Ben Mokhtar	LIRIS, CNRS, France
Salima Benbernou	Université Paris Descartes, France
Reda Bendraou	LIP6 Paris Universit�as, France
Djamal Benslimane	University of Lyon, France
Sami Bhiri	Telecom SudParis, France
Domenico Bianculli	University of Luxembourg, Luxembourg
Frederique Biennier	INSA of Lyon, France
Walter Binder	University of Lugano, Switzerland
M. Brian Blake	University of Miami, USA
Omar Boucelma	Aix-Marseille University, France
Athman Bouguettaya	RMIT, Australia
Ivona Brandic	Vienna University of Technology, Austria
Christoph Bussler	Oracle Corporation, USA
Cristina Cabanillas	Vienna University of Economics and Business, Austria
Manuel Carro	UPM and IMDEA Software Institute, Spain
Wing-Kwong Chan	City University of Hong Kong, Hong Kong, SAR China
Francois Charoy	University of Lorraine, France
Sanjay Chaudhary	Ahmedabad University, India
Shiping Chen	CSIRO ICT, Australia

Lawrence Chung	The University of Texas at Dallas, USA
Hoa Khanh Dam	University of Wollongong, Australia
Florian Daniel	University of Trento, Italy
Bruno Defude	Telecom Sud Paris, France
Shuiguang Deng	Zhejiang University, China
Nirmit Desai	IBM Research, India
Khalil Drira	LAAS Toulouse, France
Yucong Duan	Hainan University, China
Abdelkarim Erradi	Qatar University, Qatar
Rik Eshuis	Eindhoven University of Technology, The Netherlands
Onyeka Ezenwoye	Georgia Regents University, USA
Marcelo Fantinato	University of Sao Paulo, Brazil
Marie-Christine Fauvet	University of Grenoble Alpes, France
Joao E. Ferreira	University of Sao Paulo, Brazil
Walid Gaaloul	Telecom SudParis, France
N.D. Gangadhar	MS Ramaiah University of Applied Sciences, India
G.R. Gangadharan	IDRBT, India
Paolo Giorgini	University of Trento, Italy
Claude Godart	University of Lorraine, France
Mohamed Graiet	ISIMM, Tunisia
Sven Graupner	HP Labs, Palo Alto, USA
Daniela Grigori	University of Paris-Dauphine, France
Adnene Guabtini	NICTA, Australia
Armin Haller	Australian National University, Australia
Jun Han	Swinburne University of Technology, Australia
Peng Han	Chongqing Academy of Science and Technology, China, China
Bernhard Holtkamp	Fraunhofer ISST, Germany
Richard Hull	IBM Research, USA
Fuyuki Ishikawa	National Institute of Informatics, Japan
Hai Jin	HUST, China
Ejub Kajan	State University of Novi Pazar, Serbia
Dimka Karastoyanova	University of Stuttgart, Germany
Raman Kazhamiakin	Fondazione Bruno Kessler, Italy
Hamamache Kheddouci	University of Lyon, France, France
Kais Klai	University of Paris 13, France
Ryan Ko	University of Waikato, New Zealand
Gerald Kotonya	Lancaster University, UK
Peep Kungas	University of Tartu, Estonia
Philippe Lalanda	Joseph Fourier University, France
Philipp Leitner	University of Zurich, Switzerland
Henrik Leopold	VU University Amsterdam, The Netherlands
Huma Mehadisa Lepakshi	Infosys, India
Frank Leymann	University of Stuttgart, Germany
Ying Li	Zhejiang University, China
Xumin Liu	Rochester Institute of Technology, USA

Alessio Lomuscio	Imperial College London, UK
Zakaria Maamar	Zayed University, United Arab Emirates
Zaki Malik	Wayne State University, USA
Ioana Manolescu	Inria, France
Jordi Marco	Universitat Politècnica de Catalunya, Spain
Massimo Mecella	Sapienza Università di Roma, Italy
Brahim Medjahed	University of Michigan - Dearborn, USA
Jan Mendling	WU Vienna, Austria
Lars Moench	University of Hagen, Germany
Marco Montali	Free University of Bozen-Bolzano, Italy
Hamid Reza Motahari-Nezhad	HP, USA
Michael Mrissa	University of Lyon, France
Nanjangud C. Narendra	M.S. Ramaiah University, India
Surya Nepal	CSIRO, Australia
Alex Norta	Tallinn University of Technology, Estonia
Srinivasa Padmanabhuni	Infosys Technologies Limited, India
Helen Paik	UNSW, Australia
Olivier Perrin	Lorraine University, France
RadhaKrishna Pisipati	Infosys Technologies Limited, India
Marco Pistore	Fondazione Bruno Kessler, Italy
Pierluigi Plebani	Politecnico di Milano, Italy
Pascal Poizat	Université Paris Ouest and LIP6, France
Artem Polyvyanyy	Queensland University of Technology, Australia
Karthikeyan Ponnalagu	IBM Research, India
Mu Qiao	IBM Almaden Research Center, USA
Lakshmish Ramaswamy	University of Georgia, USA
Manfred Reichert	University of Ulm, Germany
Wolfgang Reisig	Humboldt-Universität zu Berlin, Germany
Stefanie Rinderle-Ma	University of Vienna, Austria
Colette Roland	Université Paris 1 Pantheon Sorbonne, France
Antonio Ruiz-Cortes	University of Sevilla, Spain
Sherif Saakr	University of New South Wales, Australia
Mohammad Sadoghi	IBM T.J. Watson Research Center, USA
Diptikalyan Saha	IBM Research, India
Iman Saleh	Intel Corporation, USA
Jun Shen	University of Wollongong, Australia
Larisa Shwartz	IBM T.J. Watson Research Center, USA
Ignacio Silva-Lepe	IBM T.J. Watson Research Center, USA
Yogesh Simmhan	Indian Institute of Science, India
Sergey Smirnov	SAP, Germany
George Spanoudakis	City University London, UK
Jianwen Su	University of California at Santa Barbara, USA
Wei Tan	IBM T.J. Watson Research Center, USA
Samir Tata	Telecom SudParis, France
Philippe Thiran	University of Namur, Belgium

Roman Vaculin	IBM T.J. Watson Research Center, USA
Guiling Wang	North China University of Technology, China
Jianwu Wang	University of California, USA
Yan Wang	Macquarie University, Australia
Zhongjie Wang	Harbin Institute of Technology, China
Ingo Weber	NICTA, Australia
Yi Wei	Microsoft, USA
Matthias Weidlich	Imperial College London, UK
Lai Xu	Bournemouth University, UK
Moez Yeddes	National Institute of Applied Sciences and Technology, Tunisia
Jian Yu	Auckland University of Technology, New Zealand
Qi Yu	Rochester Institute of Technology, USA
Uwe Zdon	University of Vienna, Austria
Weiliang Zhao	University of Wollongong, Australia
Yan Zheng	Aalto University/Xidian University, Finland
Floriano Zini	Free University of Bozen-Bolzano, Italy
Andrea Zisman	City University London, UK

Demonstration Track Committee

Adnene Guabtni	NICTA, Australia
Armin Haller	CSIRO, Australia
Athman Bouguettaya	RMIT University, Australia
Dickson Chiu	The University of Hong Kong, Hong Kong, SAR China
Djamal Benslimane	University of Lyon, France
Florian Daniel	University of Trento, Italy
Helen Paik	University of New South Wales, Australia
Ivona Brandic	Vienna University of Technology, Austria
Mohammad Sadoghi	IBM Research, USA
Philipp Leitner	University of Zurich, Switzerland
Philippe Lalanda	Joseph Fourier University, France
Pierluigi Plebani	Politecnico di Milano, Italy
Raman Kazhamiakin	SOA Research Unit, Fondazione Bruno Kessler, Trento, Italy
Sonia Ben Mokhtar	LIRIS, CNRS, France
Uwe Zdon	University of Vienna, Austria
Wei Tan	IBM T.J. Watson Research Center, USA
Xumin Liu	Rochester Institute of Technology, USA

Additional Reviewers

Nariman Ammar	Wayne State University, USA
Nour Assy	Telecom SudParis, France
Daniel Batista	University of Sao Paulo, Brazil

Oscar Cabrera Bejar	Polytechnic University of Catalunya, Spain
Lubomir Bulej	Università della Svizzera Italiana, Switzerland
Isaac Caicedo-Castro	University of Grenoble, France
Sivadon Chaisiri	University of Waikato, New Zealand
Martina De Sanctis	Fondazione Bruno Kessler, Italy
Boris Duedder	Technical University of Dortmund, Germany
Walid Fdhila	University of Vienna, Austria
Pablo Fernandez	University of Seville, Spain
Manuel Gall	University of Vienna, Austria
Genady Grabarnik	St. John's University, USA
Ikbel Guidara	LAAS-CNRS and Université de Toulouse, France
Antonio Manuel Gutierrez	University of Seville, Spain
Emna Hachicha	Telecom SudParis, France
Khayyam Hashmi	Wayne State University, USA
Regina Hebig	University of Pierre and Marie Curie, France
Tobias Hildebrandt	University of Vienna, Austria
Georg Kaes	University of Vienna, Austria
Ryan Ko	University of Waikato, New Zealand
Fabio Kon	University of Sao Paulo, Brazil
Rafael Liberato	Federal University of Technology-Parana, Brazil
Annapaola Marconi	Fondazione Bruno Kessler, Italy
Andrea Marrella	Sapienza University of Rome, Italy
Wafa Mekki	University of Sfax, Tunisia
Emna Mezghani	Luxembourg Institute of Science and Technology, Luxembourg
Erfan Najmi	Wayne State University, USA
Marcio K. Oikawa	Federal University of ABC, Brazil
Jos Antonio Parejo	University of Seville, Spain
Luca Piras	TrentoRISE, Italy
Manuel Resinas	University of Seville, Spain
Gerald Schermann	University of Zurich, Switzerland
Joel Scheuner	University of Zurich, Switzerland
Stefan Schnig	University of Bayreuth, Germany
Andr Luis Schwert	Federal University of Technology-Parana, Brazil
Sana Sellami	Aix Marseille University, France
Sebastiano Spicuglia	IBM Zurich Research Lab, Switzerland
Liang Tang	LinkedIn, USA
Giuseppe Valetto	Fondazione Bruno Kessler, Italy
Peifeng Yin	IBM Almaden Research Center, USA
Karn Yongsiriwit	Telecom SudParis, France
Mo Yu	Pennsylvania State University, USA

Contents

Internet of Services/Things

Combining Practical and Dialectical Commitments for Service Engagements	3
<i>Pankaj R. Telang, Anup K. Kalia, John F. Madden, and Munindar P. Singh</i>	
Positron: Composing Commitment-Based Protocols	19
<i>Scott N. Gerard, Pankaj R. Telang, Anup K. Kalia, and Munindar P. Singh</i>	
Analysis of Timing Constraints in Heterogeneous Middleware Interactions . . .	36
<i>Ajay Kattepur, Nikolaos Georgantas, Georgios Bouloukakis, and Valérie Issarny</i>	
Context-Driven Assessment of Provider Reputation in Composite Provision Scenarios	53
<i>Lina Barakat, Phillip Taylor, Nathan Griffiths, and Simon Miles</i>	

Data Services and Cloud Platform Management

Runtime Model-Based Privacy Checks of Big Data Cloud Services	71
<i>Eric Schmieders, Andreas Metzger, and Klaus Pohl</i>	
Optimizing Workload Category for Adaptive Workload Prediction in Service Clouds	87
<i>Chunhong Liu, Yanlei Shang, Li Duan, Shiping Chen, Chuanchang Liu, and Junliang Chen</i>	
On Developing and Operating of Data Elasticity Management Process	105
<i>Tien-Dung Nguyen, Hong-Linh Truong, Georgiana Copil, Duc-Hung Le, Daniel Moldovan, and Schahram Dustdar</i>	

Cloud Services Management

Supporting Cloud Service Operation Management for Elasticity	123
<i>Georgiana Copil, Hong-Linh Truong, and Schahram Dustdar</i>	
rSLA: Monitoring SLAs in Dynamic Service Environments	139
<i>Heiko Ludwig, Katerina Stamou, Mohamed Mohamed, Nagapramod Mandagere, Bryan Langston, Gabriel Alatorre, Hiroaki Nakamura, Obinna Anya, and Alexander Keller</i>	

AISLE: Assessment of Provisioned Service Levels in Public IaaS-Based Database Systems 154
Jörn Kuhlenkamp, Kevin Rudolph, and David Bermbach

Service Composition

Are RESTful APIs Well-Designed? Detection of their Linguistic (Anti) Patterns 171
Francis Palma, Javier Gonzalez-Huerta, Naouel Moha, Yann-Gaël Guéhéneuc, and Guy Tremblay

Aggregating Functionality, Use History, and Popularity of APIs to Recommend Mashup Creation 188
Aditi Jain, Xumin Liu, and Qi Yu

Integrating Gaussian Process with Reinforcement Learning for Adaptive Service Composition 203
Hongbing Wang, Qin Wu, Xin Chen, and Qi Yu

Scalable SaaS-Based Process Customization with *CaseWalls* 218
Yu-Jen John Sun, Moshe Chai Barukh, Boualem Benatallah, and Seyed-Mehdi-Reza Beheshti

Business Process Management

Correlation Mining: Mining Process Orchestrations Without Case Identifiers. 237
Shaya Pourmirza, Remco Dijkman, and Paul Grefen

Verification of GSM-Based Artifact-Centric Systems by Predicate Abstraction. 253
Pavel Gonzalez, Andreas Griesmayer, and Alessio Lomuscio

Mining and Querying Process Change Information Based on Change Trees. 269
Georg Kaes and Stefanie Rinderle-Ma

Property Preservation in Adaptive Case Management. 285
Rik Eshuis, Richard Hull, and Mengfei Yi

Cloud Services (Short Papers)

Modelling and Optimizing Bandwidth Provision for Interacting Cloud Services 305
Chao Chen, Ligang He, Bo Gao, Cheng Chang, Kenli Li, and Keqin Li

Four-Fold Auto-Scaling on a Contemporary Deployment Platform
Using Docker Containers 316
*Philipp Hoenisch, Ingo Weber, Stefan Schulte, Liming Zhu,
and Alan Fekete*

An SLA-Based Advisor for Placement of HPC Jobs on Hybrid Clouds 324
*Kiran Mantripragada, Leonardo P. Tizzei, Alecio P.D. Binotto,
and Marco A.S. Netto*

Optimizing Long-Term IaaS Service Composition 333
Sajib Mistry, Athman Bouguettaya, Hai Dong, and A.K. Qin

QoS and Trust (Short Papers)

On the Complexity of QoS-Aware Service Selection Problem. 345
*Faisal N. Abu-Khzam, Cristina Bazgan, Joyce El Haddad,
and Florian Sikora*

TRACE: A Dynamic Model of Trust for People-Driven Service
Engagements: Combining Trust with Risk, Commitments, and Emotions 353
Anup K. Kalia, Pradeep K. Murukannaiah, and Munindar P. Singh

A Context-Aware Approach for Personalised and Adaptive QoS
Assessments 362
Lina Barakat, Adel Taweel, Michael Luck, and Simon Miles

Service Composition (Short Papers)

Spatio-Temporal Composition of Crowdsourced Services 373
Azadeh Ghari Neiat, Athman Bouguettaya, and Timos Sellis

Design for Adaptation of Distributed Service-Based Systems 383
*Antonio Bucchiarone, Martina De Sanctis, Annapaola Marconi,
Marco Pistore, and Paolo Traverso*

Industry Track Papers

Automatic Deployment of Services in the Cloud with Aeolus Blender 397
*Roberto Di Cosmo, Antoine Eiche, Jacopo Mauro, Stefano Zacchiroli,
Gianluigi Zavattaro, and Jakub Zwolakowski*

Analyzing Resource Behavior to Aid Task Assignment in Service Systems 412
Renuka Sindhgatta, Aditya Ghose, and Gaargi Banerjee Dasgupta

SenseX: Design and Deployment of a Pervasive Wellness Monitoring Platform for Workplaces	427
<i>Rakshit Wadhwa, Amandeep Chugh, Abhishek Kumar, Mridula Singh, Kuldeep Yadav, Sharanya Eswaran, and Tridib Mukherjee</i>	
Opportunities for Process Improvement: A Cross-Clientele Analysis of Event Data Using Process Mining	444
<i>R.P. Jagadeesh Chandra Bose, Avantika Gupta, Deepthi Chander, Ajith Ramanath, and Koustuv Dasgupta</i>	
Pricing IT Services Deals: A More Agile Top-Down Approach	461
<i>Aly Megahed, Kugamoorthy Gajananan, Mari Abe, Shun Jiang, Mark Smith, and Taiga Nakamura</i>	
Demonstration Track Papers	
SimMon: A Toolkit for Simulating Monitoring Mechanism in Cloud Computing Environments	477
<i>Xinkui Zhao, Jianwei Yin, Pengxiang Lin, Chen Zhi, Shichun Feng, Hao Wu, and Zuoning Chen</i>	
CASE: A Platform for Crowdsourcing Based API Search.	482
<i>Tingting Liang, Liang Chen, Zhining Xie, Wei Yang, and Jian Wu</i>	
WSTP: Web Services Tagging Platform.	486
<i>Sana Sellami and Hanane Becha</i>	
Personalized Messaging Engine: The Next Step in Employee Engagement . . .	491
<i>Varun Sharma, Abhishek Tripathi, Saurabh Srivastava, Aditya Hegde, and Koustuv Dasgupta</i>	
Offering Context-Aware Personalised Services for Mobile Users	495
<i>Marie-Christine Fauvet, Sanjay Kamath, Isaac-Bernardo Caicedo-Castro, Pathathai Na-Lumpoon, Ahmed Lbath, and Lorraine Goeuriot</i>	
Author Index	499

Internet of Services/Things

Combining Practical and Dialectical Commitments for Service Engagements

Pankaj R. Telang¹(✉), Anup K. Kalia², John F. Madden³,
and Munindar P. Singh²

¹ Cisco Systems, Research Triangle Park, Durham, NC 27709, USA
ptelang@gmail.com

² North Carolina State University, Raleigh, NC 27695-8206, USA
{akkalia,singh}@ncsu.edu

³ Duke University Medical Center, Durham, NC 27710, USA
john.madden@duke.edu

Abstract. We understand a service engagement as a form of collaboration arising in a sociotechnical system (STS). Although STSs are fruitfully modeled using normative abstractions such as commitments, a conventional (practical) commitment can capture only part of the story, namely, a debtor’s promise to the creditor to bring about the consequent if the antecedent holds. In contrast, in a dialectical commitment, which we highlight, a debtor asserts to the creditor that the consequent is true if the antecedent is. For example, a customer may dialectically commit to a seller that the product she received is damaged but may not practically commit to damaging the product. We introduce a novel bipartite operationalization of dialectical commitments that separates their objective and subjective aspects and thus avoids the problems arising if we merely treat dialectical like practical commitments. We express that operationalization in temporal logic, developing a verification tool based on NuSMV, a well-known model-checker, to verify if the participants’ interactions comply with the participants’ dialectical commitments. We present a set of modeling patterns that incorporate both practical and dialectical commitments. We validate our proposal using a real-world scenario of contradictory medical diagnoses by different specialists.

1 Introduction

A service engagement involves two or more autonomous parties interacting with each other and is thus a prototypical sociotechnical system (STS) [22]. An STS can be fruitfully modeled using normative relationships. To this end, commitments have been extensively employed in modeling service engagements (and associated business processes) [5, 19, 20, 24]. A key benefit of commitments over traditional approaches is that commitments capture outcomes in a declarative manner and minimally constrain the behavior of the participants. Two kinds of commitment are known in the literature [13, 16, 21]: practical and dialectical. In a practical commitment, a debtor agent promises a creditor agent to bring about a condition (consequent) if some other condition (antecedent) holds. For example, a customer may commit to paying a reseller if the reseller delivers the goods.

In a dialectical commitment, the debtor claims that the consequent holds provided the antecedent does. For example, a customer may dialectically commit to a reseller that the customer received the goods in a damaged condition. These commitments differ in the nature of their standard of satisfaction. For example, a customer may dialectically commit that it received damaged goods, but may not practically commit to damaging the goods. Previous research has nearly always considered only practical commitments [6, 25, 26], a recent exception being Baldoni et al. [2].

The present paper incorporates dialectical commitments in modeling an STS to tackle a previously ignored challenge, namely, how participants make claims about putative facts, claims that may be mutually inconsistent. For example, a customer may claim that goods received are damaged whereas the courier may claim the goods delivered were not damaged. Although this paper doesn't tackle norm types other than commitments [22], by bringing forth dialectical commitments, it supports the possibility of modeling disputes between participants in STSs and disparities in their policies. This paper sheds light on how potentially to resolve such disputes, thereby facilitating policy-governed secure collaboration.

Research Question and Contributions. When we model secure collaboration in STSs in normative terms [23], it is important to accommodate disputes among STS participants regarding facts and norms. Previous approaches include the objective, but omit the social, aspect of norms in their lifecycles [22]. This leads to our research question: How can we formalize norms in a manner that incorporates their objective and subjective elements and supports verification of interactions on comprehensive grounds?

This paper is restricted to two norm types: dialectical and practical commitments. It contributes a novel operational model and temporal logic formalization based on Computational Tree Logic (CTL) along with a tool based on NuSMV [17], a CTL model checker, to verify if participants' interactions comply with their commitments. This paper provides a set of modeling patterns incorporating practical and dialectical commitments. We evaluate our approach on a breast cancer diagnosis process specified by a committee of experts called by a major government agency (Office of the Assistant Secretary for Planning and Evaluation (ASPE), US Department of Health and Human Services) [1]. The significance of this work arises from its expanding the operational treatment and formal verification of commitments to incorporate dialectical commitments, thereby enabling new applications that previous approaches cannot tackle.

2 Background

We illustrate the generality of our approach by introducing it via Cisco's Quote to Cash (QTC) business process [25] and evaluating it via a healthcare collaboration scenario (introduced in Sect. 4). The QTC process encompasses all of the key activities that begin from a customer requesting a quote, and end in Cisco receiving payment from the customer. The participants in this process include

customers, resellers, distributors, logistics providers, banks, contract manufacturers, and service providers. A customer purchases goods either directly from Cisco, or from a reseller. In addition to selling the goods, a reseller provides value-added services of installing and configuring goods. A reseller purchases goods either from a distributor, or from Cisco. A distributor always purchases goods from Cisco. Unlike a reseller, a distributor may purchase and stock the goods in its warehouse. To build and ship its products, Cisco uses contract manufacturers and transportation providers respectively. The participants use different banks and credit companies for making payments.

2.1 Practical Commitments

A practical commitment [21] $C(\text{DEBTOR}, \text{CREDITOR}, \text{O-CONTEXT}, \text{antecedent}, \text{consequent})$ means that DEBTOR commits to CREDITOR in the organizational context O-CONTEXT to bring about the consequent provided the antecedent holds. (For brevity, we omit O-CONTEXT where appropriate.) For example, $C(\text{CISCO}, \text{CUSTOMER}, \text{COURT}, \text{pay}, \text{deliver goods})$ means that CISCO commits under the o-context COURT to CUSTOMER to deliver goods, provided CUSTOMER pays.

We describe the lifecycle of a practical commitment from Fig. 1 [25] using the above example. When CISCO creates the commitment, its state changes to active from null. If CUSTOMER pays CISCO (antecedent holds), the commitment is detached. The commitment is terminated if CISCO cancels the commitment when conditional, or CUSTOMER releases CISCO from the commitment. The commitment is satisfied when the goods are delivered (consequent holds). It is violated if CUSTOMER has paid up (antecedent holds), but CISCO does not deliver the goods (consequent fails), or if CISCO cancels the commitment. When the commitment is conditional, if CUSTOMER does not pay (antecedent fails) CISCO, then the commitment expires. If CISCO delegates the commitment to another company (delegatee), CISCO may suspend the commitment, making it pending. If the delegatee company fails to provide goods to CUSTOMER, then CISCO may reactivate its commitment to CUSTOMER.

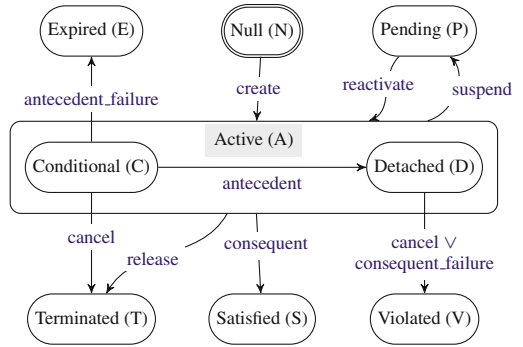


Fig. 1. Practical commitment lifecycle as a state transition diagram.

2.2 Computation Tree Logic

Computation Tree Logic (CTL) [4] is a temporal logic based on a branching time structure. Each temporal operator in CTL has two components. The first

component is a path quantifier: either **A**, meaning on all of the paths; or **E**, meaning on at least one path. The second component is a linear-time operator: **F**, meaning in a future state; **G**, meaning (globally) in all future states; and **X**, meaning in the next state. A CTL formula may contain the standard logical operators: \neg , \wedge , \vee , and \rightarrow , meaning negation, conjunction, disjunction, and implication, respectively. As an example, $AG(p \rightarrow AFq)$ means that on all paths if proposition p holds in a state, then on all paths emanating from that state proposition q holds in a future state.

3 Dialectical Commitments

The lifecycle of a practical commitment, as shown in Fig. 1, is inadequate for capturing the semantics of a dialectical commitment. For example, consider Fig. 2, which shows a possible execution in which CUSTOMER and CISCO interact to decide if the goods delivered to CUSTOMER are damaged. CUSTOMER informs CISCO, i.e., dialectically commits that the goods are damaged.

However, CISCO disagrees with CUSTOMER and challenges CUSTOMER's claim. That is CISCO dialectically commits that the goods are not damaged. CUSTOMER then requests a relevant higher authority, such as COURT, for resolution. COURT concludes that the goods are not damaged. CUSTOMER agrees with COURT and retracts its dialectical commitment.

If we employ the lifecycle of a practical commitment to handle CUSTOMER's dialectical commitment, then the commitment is violated since COURT concludes that the goods are not damaged. Thus, the lifecycle of a practical commitment fails to handle CUSTOMER's retraction of a dialectical commitment appropriately.

We write a dialectical commitment using a notation similar to that of a practical commitment: $D(\text{DEBTOR}, \text{CREDITOR}, \text{O-CONTEXT}, \text{antecedent}, \text{consequent})$. For example, $D(\text{CUSTOMER}, \text{CISCO}, \text{COURT}, \top, \text{goods-damaged})$ means that CUSTOMER dialectically and unconditionally (antecedent is \top , true) commits to CISCO that the goods are damaged. We allow the debtor to be a set of roles. For example, in the QTC process, CUSTOMER and RESELLER may jointly commit to CISCO that the goods are damaged: $D(\{\text{CUSTOMER}, \text{RESELLER}\}, \text{CISCO}, \text{COURT}, \top, \text{goods-damaged})$.

3.1 The Proposed Lifecycle of Dialectical Commitments

Figure 3 shows our proposed lifecycle of a dialectical commitment. The state of a dialectical commitment has two dimensions: objective (computed based on

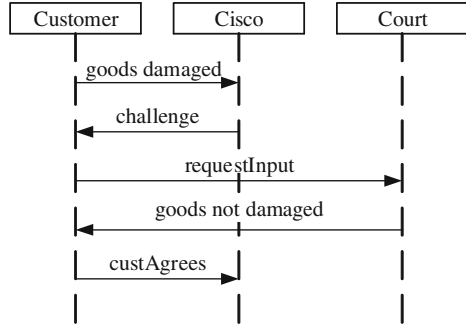


Fig. 2. Customer and Cisco interactions.

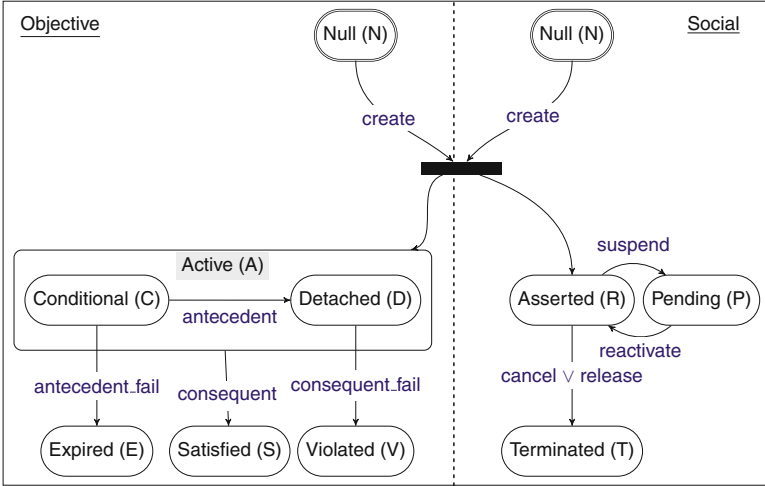


Fig. 3. Dialectical commitment lifecycle as a state-transition diagram.

the antecedent and consequent, treated as objective facts) and social (computed based on the creditor or the debtor’s actions). Thus, the state of a dialectical commitment is bipartite and written as a pair, e.g., $\langle satisfied, asserted \rangle$.

A dialectical commitment is null before it is created. Upon creation, its objective state becomes active and its social state becomes asserted. The active state has two substates: conditional and detached. The commitment becomes detached when its antecedent holds. If the antecedent of a conditional commitment fails, then the commitment becomes expired. The commitment is satisfied if its consequent becomes true when it is active. The commitment becomes violated if its consequent fails when it is detached. On the social side, if the debtor cancels or suspends the commitment when it is asserted, it becomes terminated or pending, respectively. If the debtor reactivates the commitment when it is pending, it becomes asserted. We write the bipartite state of a dialectical commitment as a pair: $\langle \text{objective-state}, \text{social-state} \rangle$. We write the objective state as $D_{\text{obj}}^{\text{ostate}}$ and social state as $D_{\text{soc}}^{\text{sstate}}$, where $\text{ostate} \in \{N, C, D, E, S, V\}$ and $\text{sstate} \in \{N, R, P, T\}$ (state labels are from Fig. 3).

We describe the progression of a dialectical commitment in the CUSTOMER-CISCO interactions from Fig. 2. CUSTOMER informs CISCO that the goods are damaged and thus creates the dialectical commitment: $D_u = D(\text{CUSTOMER}, \text{CISCO}, \text{COURT}, \top, \text{goods-damaged})$. Upon creation, D_u ’s state is $\langle detached, asserted \rangle$ (*detached* since its antecedent is true (\top)).

However, CISCO disagrees with CUSTOMER and challenges CUSTOMER’s claim, thus creating the dialectical commitment: $D_c = D(\text{CISCO}, \text{CUSTOMER}, \text{COURT}, \top, \neg\text{goods-damaged})$. Upon creation, D_c is $\langle detached, asserted \rangle$. CUSTOMER and CISCO may resolve their difference of opinion among themselves. But in Fig. 2 they escalate the dispute to COURT on the condition of goods. COURT concludes that the goods are not damaged, which causes D_u to transition to $\langle violated,$

asserted), and D_c to transition to $\langle \textit{satisfied}, \textit{asserted} \rangle$. Finally, CUSTOMER agrees that the goods are not damaged, that is, CUSTOMER cancels D_u , causing its state to transition to $\langle \textit{violated}, \textit{terminated} \rangle$.

3.2 Formalization

We now formalize the lifecycle of dialectical commitments in CTL. We group the specifications into four groups: state-action, state-state, terminal states, and acceptable executions. To save the space, we describe one from each group.

State-Action Transitions. The CTL specifications for state-action transitions follow from the lifecycle given above. For brevity, we explain only a few of them in English.

- SA1. $AG (D_{obj}^N \wedge \textit{create} \wedge \neg \textit{antecedent} \rightarrow AX D_{obj}^C)$
- SA2. $AG (D_{obj}^N \wedge \textit{create} \wedge \textit{antecedent} \rightarrow AX D_{obj}^D)$
- SA3. $AG (D_{obj}^C \wedge \textit{antecedent} \rightarrow AX D_{obj}^D)$
- SA4. $AG (D_{obj}^C \wedge \textit{antecedent_fail} \rightarrow AX D_{obj}^E)$
- SA5. $AG (D_{obj}^D \wedge \textit{consequent_fail} \rightarrow AX D_{obj}^V)$
- SA6. $AG (D_{obj}^{CV} \wedge \textit{consequent} \rightarrow AX D_{obj}^S)$

On any path, if a dialectical commitment is **conditional** or **DETACHED** in a state and its consequent holds, then on all paths emanating from that state in the next state, the commitment's objective state becomes **satisfied**.

- SA7. $AG (D_{soc}^N \wedge \textit{create} \rightarrow AX D_{soc}^R)$
On any path, if a dialectical commitment's social state is **null** in a state and the debtor creates it, then on all paths emanating from that state in the next state, the commitment's social state becomes **asserted**.
- SA8. $AG (D_{soc}^R \wedge \textit{suspend} \rightarrow AX D_{soc}^P)$
- SA9. $AG (D_{soc}^P \wedge \textit{reactivate} \rightarrow AX D_{soc}^R)$
- SA10. $AG (D_{soc}^R \wedge (\textit{cancel} \vee \textit{release}) \rightarrow AX D_{soc}^T)$

SA1 means that on any path, if a dialectical commitment is **null** in a state, the antecedent is not holding, and the debtor creates it, then on all paths emanating from that state, the commitment objectively becomes **conditional** in the next state.

State-State Transitions. These follow from the dialectical commitment lifecycle.

- SS1. $AG (D_{obj}^N \rightarrow AX D_{obj}^{N \vee C \vee E \vee D \vee S \vee V})$
- SS2. $AG (D_{obj}^C \rightarrow AX D_{obj}^{C \vee E \vee D \vee S})$
- SS3. $AG (D_{obj}^D \rightarrow AX D_{obj}^{D \vee V \vee S})$
- SS4. $AG (D_{soc}^N \rightarrow AX D_{soc}^{N \vee R})$
- SS5. $AG (D_{soc}^R \rightarrow AX D_{soc}^{R \vee T \vee P})$
- SS6. $AG (D_{soc}^P \rightarrow AX D_{soc}^{P \vee R})$

SS1 means if a dialectical commitment is objectively **null** in a state, then on all paths emanating from that state, in the next state, the commitment may objectively remain **null** or may transition to **conditional**, **expired**, **detached**, **satisfied**, or **violated**.

Terminal States. These follow from the dialectical commitment lifecycle.

$$\begin{array}{ll} \text{TS1. } \text{AG} (D_{obj}^E \rightarrow \text{AX } D_{obj}^E) & \text{TS2. } \text{AG} (D_{obj}^S \rightarrow \text{AX } D_{obj}^S) \\ \text{TS3. } \text{AG} (D_{obj}^V \rightarrow \text{AX } D_{obj}^V) & \text{TS4. } \text{AG} (D_{soc}^T \rightarrow \text{AX } D_{soc}^T) \end{array}$$

TS1 means on any path, if a dialectical commitment is objectively expired in a state, then on all paths emanating from that state in the next state, the commitment objectively remains expired.

Acceptable Executions. The above CTL specifications, which follow from the lifecycle, represent hard integrity requirements on the executions. The participants may have additional requirements on acceptable executions. We now describe some common acceptable executions.

$$\begin{array}{ll} \text{AE1. } \text{AF AG} (D_{obj}^N \vee D_{obj}^C) & \text{AE2. } \text{AF AG} (D_{obj}^E \wedge D_{soc}^R) \\ \text{AE3. } \text{AF AG} (D_{obj}^S \wedge D_{soc}^R) & \text{AE4. } \text{AF AG} (D_{obj}^V \wedge D_{soc}^T) \end{array}$$

AE1 means an execution is acceptable if a dialectical commitment is never created or remains forever conditional on it. AE2 means an execution is acceptable if a dialectical commitment is created but later expires. AE3 means on an execution, a dialectical commitment may be objectively satisfied and socially asserted, i.e., $\langle \textit{satisfied}, \textit{asserted} \rangle$. However, such an execution may be acceptable since the debtor is asserting a statement that is deemed objectively true. AE4 means on an execution, a debtor may create a dialectical commitment whose consequent turns out to be false, that is, the commitment transitions to: $\langle \textit{violated}, \textit{asserted} \rangle$. In such a case, the debtor should cancel the commitment thus transitioning its state to: $\langle \textit{violated}, \textit{terminated} \rangle$. Debtor's cancellation implies that the debtor acknowledges its error. In some scenarios, debtor may be penalized for such fallacies—the context may create a commitment in which the debtor is required to pay a penalty to the creditor.

The CTL specification capturing the above desirable states of a dialectical commitment is: $\text{AF AG} (D_{obj}^N \vee D_{obj}^C \vee (D_{obj}^S \wedge D_{soc}^R) \vee (D_{obj}^V \wedge D_{soc}^T))$. This specification means that on all paths in the future, a dialectical commitment's objective state remains *null* or *conditional*, or its objective and social state becomes $\langle \textit{satisfied}, \textit{asserted} \rangle$, or $\langle \textit{violated}, \textit{terminated} \rangle$.

These examples pertain to executions ending up in certain states. In some cases, the participants may desire executions that pass through some intermediate states. We can state and verify additional properties on intermediate states as well. For example, we can write the requirement that D should always be created as: $\text{AF } D_{obj}^{C \vee D}$.

3.3 Modeling Patterns

This section presents a nonexhaustive set of representative modeling patterns.

Service Provisioning with Claimed Correctness. A provider (1) practically commits to a client to bring about a consequent condition if some antecedent condition holds, and (2) dialectically commits that either the client would agree

with the consequent, or in case of a disagreement between the client and the provider, a higher authority would agree with the consequent.

$$\begin{aligned} C_1 &= C(\text{PROVIDER}, \text{CLIENT}, \text{ant}, \text{con}) \\ D_1 &= D(\text{PROVIDER}, \text{CLIENT}, \text{con}, \text{clientAgrees} \vee \text{authAgrees}) \end{aligned}$$

For example, RESELLER (practically) commits to CUSTOMER to providing and installing the goods if CUSTOMER pays: $C(\text{RESELLER}, \text{CUSTOMER}, \text{pay}, \text{goods} \wedge \text{INSTALL})$. And RESELLER dialectically commits to CUSTOMER that the goods will be in a working condition, and the installation service acceptable: $D(\text{RESELLER}, \text{CUSTOMER}, \text{goods}, \text{clientGoodsWorking} \vee \text{authGoodsWorking})$, $D(\text{RESELLER}, \text{CUSTOMER}, \text{install}, \text{clientAcceptableInstallation} \vee \text{authAcceptableInstallation})$.

Escalation. O-CONTEXT commits to bringing about the creation of a commitment (C_2) that if the PROVIDER violates its commitment (C_1), and the CLIENT escalates the (presumed) violation to the O-CONTEXT. In C_3 , another PROVIDER commits to CLIENT to bring about the consequent. Additionally, the O-CONTEXT may penalize the violating PROVIDER or not, depending on the modeled settings and the particular circumstances that obtain, some of which need not concern CLIENT.

$$\begin{aligned} C_1 &= C(\text{PROVIDER}, \text{CLIENT}, \text{ant}, \text{con}) \\ C_2 &= C(\text{o-context}, \text{CLIENT}, \text{vio}(C_1) \wedge \text{escalate}, \text{create}(C_3)) \\ C_3 &= C(\text{PROVIDER}', \text{CLIENT}, \text{ant}', \text{con}) \end{aligned}$$

For example, DISTRIBUTOR practically commits to delivering goods to CUSTOMER: $C_1 = C(\text{DISTRIBUTOR}, \text{CUSTOMER}, \top, \text{goods})$. If DISTRIBUTOR fails to deliver the goods, the context COURT directs another distributor to deliver the goods: $C_2 = C(\text{COURT}, \text{CUSTOMER}, \text{vio}(C_1) \wedge \text{escalate}, \text{create}(C_3))$, $C_3 = C(\text{DISTRIBUTOR}', \text{CUSTOMER}, \top, \text{goods})$.

Chained Service Provisioning with Jointly Claimed Correctness.

Provider SP1 commits to a client to bring about a consequent if some antecedent holds. Additionally, SP1 dialectically commits that either the client or (in case of a disagreement between the client and the provider) a higher authority would agree that the consequent holds, if providers SP2 and SP3 do not violate their dialectical commitment (D_2). SP2 and SP3 jointly dialectically commit to SP1 that either SP1 or (in case of a disagreement) a higher authority would agree that con-3 holds.

$$\begin{aligned} C_1 &= C(\text{SP1}, \text{CLIENT}, \text{ant1}, \text{con1}) \\ D_1 &= D(\text{SP1}, \text{CLIENT}, \neg\text{vio}(D_2) \wedge \text{con1}, \text{clientAgreeCon1} \vee \text{authAgreeCon1}) \\ C_2 &= C(\text{SP3}, \text{SP2}, \text{ant2}, \text{con2}) \\ C_3 &= C(\text{SP2}, \text{SP1}, \text{ant3}, \text{con3}) \\ D_2 &= C(\{\text{SP2}, \text{SP3}\}, \text{SP1}, \text{con3}, \text{sp1AgreeCon3} \vee \text{authAgreeCon3}) \end{aligned}$$

4 Evaluation

We evaluate our approach on a breast cancer diagnosis process specified by a committee of experts called by a major government agency (US Department of Health and Human Services) [1]. This process models five roles: PATIENT, PHYSICIAN, RADIOLOGIST, PATHOLOGIST, and REGISTRAR. The roles interact as follows: (1) the physician orders a mammography (imaging) exam for the patient; (2) if the radiologist notices suspicious calcifications, she recommends a biopsy; (3) if the physician agrees, she performs a biopsy, and sends the collected tissue specimen to the pathologist; (4) the pathologist analyzes the specimen, and performs ancillary studies; (5) the pathologist and radiologist may confer to reconcile their results and produce a consensus report; (6) the physician reviews the integrated report with the patient to create a treatment plan; and (7) the pathologist forwards his report to a cancer registry’s registrar.

We apply the patterns on the cancer diagnosis scenario to produce a commitment-based model. We rename the pattern roles with the scenario-specific role names, and substitute the scenario-specific tasks as the antecedents and consequents of the appropriate commitments. We describe the commitments shown in Table 1 and the patterns that compose the model.

Table 1. Commitment-based model for the diagnosis process

C ₁	C(PHY, PAT, diagReq \wedge \neg vio(C ₂) \wedge \neg vio(C ₃), diag)
C ₂	C(PAT, PHY, iApptReq, iApptKept)
C ₃	C(PAT, PHY, bApptReq, bApptKept)
C ₄	C(RAD, PHY, biopsyReq \wedge bApptKept, radPathResults)
C ₅	C(RAD, PHY, imagingReq \wedge iApptKept, imagingResults)
C ₆	C(PATH, RAD, pathologyReq \wedge tissue, pathResults)
C ₇	C(PATH, HOSP, patHasCancer, patRepToRegistrar)
C ₈	C(REG, HOSP, patRepToRegistrar, addPatToRegistry)
C ₉	C(HOSP, PHY, vio(C ₅) \wedge esc, create(C ₅ ') \wedge create(D ₂ '))
C ₁₀	C(BOARD, RAD, radReq, BAgreesPath \vee BDisagreesPath)
C ₁₁	C(BOARD, PHY, phyReq, BAgreesRad \vee BDisagreesRad)
C ₁₂	C(BOARD, PAT, patReq, BAgreesPhy \vee BDisagreesPhy)
D ₁	D(PHY, PAT, diag \wedge \neg vio(D ₃), patAgrees \vee BAgreesDiag)
D ₂	D(RAD, PHY, imaging, phyAgreesI \vee BAgreesI)
D ₃	D({RAD, PATH}, PHY, radPathResults, phyAgreesRP \vee BAgreesRP)
D ₄	D(RAD, PATH, radResults, pathAgreesR \vee BAgreesR)
D ₅	D(PATH, RAD, pathResults, radAgreesP \vee BAgreesP)

(PHY: PHYSICIAN, PAT: PATIENT, RAD: RADIOLOGIST, BOARD: TUMOR BOARD, REG: REGISTRAR, HOSP: HOSPITAL)

Patient’s Appointments. Practical commitments (C_2, C_3). PATIENT commits to PHYSICIAN to keep her imaging (C_2) and biopsy appointments (C_3) if requested.

Add Patient to Registry. Practical commitments (C_7, C_8). PATHOLOGIST commits to HOSPITAL (C_7) to reporting PATIENT to REGISTRAR if PATIENT has cancer, and REGISTRAR commits to HOSPITAL (C_8) to add PATIENT to the registry.

Patient’s Radiology and Pathologist’s Diagnosis. Chained service provider with jointly claimed correctness (C_1, D_1, C_4, C_6, D_3). PATHOLOGIST commits to RADIOLOGIST (C_6) to provide a pathology report if RADIOLOGIST requests it and provides a tissue sample. RADIOLOGIST commits to PHYSICIAN (C_4) to provide an integrated radiology and pathology report if PHYSICIAN requests it and PATIENT keeps the necessary appointment. PATHOLOGIST and RADIOLOGIST jointly dialectically commit to PHYSICIAN (D_3) regarding the correctness of the integrated report. PHYSICIAN commits to PATIENT (C_1) to provide a diagnosis report if PATIENT requests it and keeps necessary appointments. PHYSICIAN dialectically commits to PATIENT (D_1) to the correctness of the diagnosis report if the integrated radiology and pathology report is correct.

Patient’s Imaging. Service provisioning with correctness (C_5, D_2). RADIOLOGIST commits to PHYSICIAN (C_5) to provide imaging results if PHYSICIAN requests the results. In addition, RADIOLOGIST dialectically commits to PHYSICIAN (D_2) regarding the correctness of the imaging results.

Escalate Radiologist’s Failure to Provide Imaging Results. Escalate (C_5, C_9, C_5', D_2'). HOSPITAL commits to PHYSICIAN to bring about the creation of practical (C_5') and dialectical (D_2') commitments from an alternative RADIOLOGIST if the original RADIOLOGIST violates commitment C_5 and PHYSICIAN escalates the violation.

Tumor Board Provides Input on a Diagnosis. Practical commitments ($C_{10}, C_{11}, C_{12}, C_{13}$). TUMOR BOARD commits to PHYSICIAN, RADIOLOGIST, PATIENT, and PATHOLOGIST to provide its input on a diagnosis upon request.

Radiologist and Pathologist Guarantee their Diagnoses. Dialectical commitments (D_4, D_5). RADIOLOGIST dialectically commits (D_4) to PATHOLOGIST that upon providing the radiology report, either PATHOLOGIST would agree with those results, or in the case of a disagreement, TUMOR BOARD will agree with those results. PATHOLOGIST makes a similar commitment (D_5) to RADIOLOGIST regarding the pathology report.

4.1 Verification

This section applies our verification approach to the ASPE process. We adopt the UML 2.0 Sequence Diagram notation [18] to create sequence diagrams for the model from Table 1. Figure 4 shows one of the sequence diagrams. The condition on the outer opt(ional) block is that RADIOLOGIST has reported the imaging

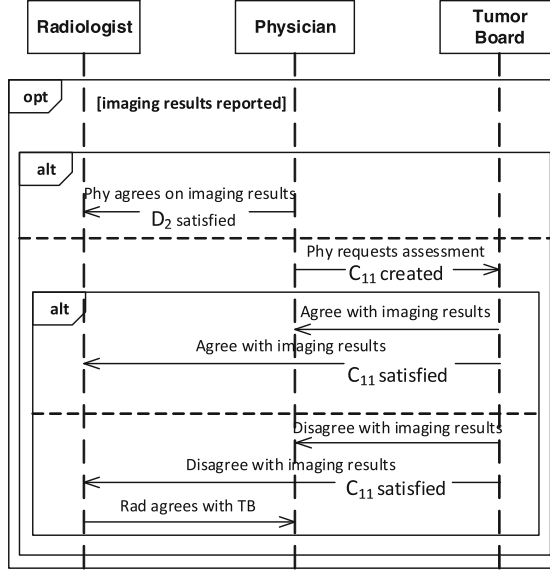


Fig. 4. PHYSICIAN requests TUMOR BOARD to review the imaging results.

results (whether PATIENT has cancer or not) to PHYSICIAN, and created D_2 . In the nested alt(ernate) block, PHYSICIAN either agrees with the imaging results, thus satisfying D_2 , or requests TUMOR BOARD for an assessment, thus creating C_{11} . In the inner alt(ernate) block, TUMOR BOARD either agrees or disagrees with the imaging results. In either case, TUMOR BOARD satisfies C_{11} . If TUMOR BOARD disagrees with the imaging results, RADIOLOGIST cancels and retracts D_2 by informing PHYSICIAN her agreement with TUMOR BOARD.

We develop a NuSMV module for dialectical commitments. We employ this module in verifying models that contain dialectical commitments. Our verification tool (based on NuSMV) [10] takes sequence diagrams and a commitment model as the input. It reports if the sequence diagrams comply with the commitments in the model.

On a computer with 2.66 GHz Intel Core 2 Duo processor, and 8 GB memory, our tool verified the set of sequence diagrams we developed for this scenario (including the one from Fig. 4) in 0.2s. Our tool reported that the sequence diagrams satisfy the model from Table 1. To demonstrate how our approach detects an error, we remove the message from RADIOLOGIST to PHYSICIAN agreeing to TUMOR BOARD’s assessment from the sequence diagram in Fig. 4. Figure 5 shows a partial screenshot of the NuSMV output demonstrating that the model fails to satisfy the (highlighted) CTL specification. The specification shows that $AF\ AG\ (D_{obj}^N \vee D_{obj}^C \vee (D_{obj}^S \wedge D_{soc}^R) \vee (D_{obj}^V \wedge D_{soc}^T))$ is false for D_2 . The counterexample shows a trace in which RADIOLOGIST violates D_2 ; i.e., D_2 remains in the state (violated, asserted). This means RADIOLOGIST does not agree with TUMOR BOARD’s recommendation, and does not cancel D_2 .

```

-- specification AG (socialstatus = TERMINATED -> AX socialstatus = TERMINATE
D) IN D2 is true
-- specification AF (AG (((objstatus = NULL | objstatus = CONDITIONAL) | (obj
status = SATISFIED & socialstatus = ASSERTED)) | (objstatus = VIOLATED & soci
alstatus = TERMINATED))) IN D2 is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  i_Collaboration1_Interaction1_alt1_if.diagnosisRequested1 = 0
  i_Collaboration1_Interaction1_alt1_if.acceptsRequest = 0
  i_Collaboration1_Interaction1_alt1_else.offersDiagnosis = 0
  i_Collaboration1_Interaction1_alt1_else.diagnosisRequested2 = 0
  i_Collaboration1_Interaction2_opt1.confirmSuspiciousLump = 0
  i_Collaboration1_Interaction2_opt1.requestsImaging = 0
  i_Collaboration1_Interaction2_opt1.agreeForImaging = 0
  i_Collaboration1_Interaction3_opt1.imagingRequested = 0
  i_Collaboration1_Interaction3_opt1.agreesForImagingR = 0
  i_Collaboration1_Interaction4_alt1_if.iAppointmentRequested = 0
  i_Collaboration1_Interaction4_alt1_if.arriveForImaging = 0
  i_Collaboration1_Interaction4_alt1_else.arriveForImaging = 0

```

Fig. 5. Tool output indicating an error in the sequence diagrams with respect to the commitments.

4.2 Benefits of Dialectical Commitments

Our approach captures relationships between the participants in terms of practical and dialectical commitments and omits the internal activities of individual participants (e.g., PATHOLOGIST’s slides activity). In this way, it avoids tight coupling between the participants. In addition, our approach provides a basis for answering some significant questions, which the traditional approach cannot answer.

What happens if the treatment plan turns out to be incorrect? Who is or are accountable? An incorrect treatment plan arises from an incorrect integrated radiology and pathology report, which means RADIOLOGIST and PATHOLOGIST both violate their joint dialectical commitment D_3 . In this case, D_1 never detaches, and thus PHYSICIAN is not accountable for the incorrect diagnosis (that is, he does not violate D_1).

What happens if RADIOLOGIST delivers the mammography results on time but her diagnosis is wrong? RADIOLOGIST violates D_2 by delivering an incorrect mammography. PHYSICIAN may incorrectly conclude that PATIENT is free of cancer. In such a case, RADIOLOGIST would be accountable for the erroneous claim.

The questions show how our approach produces models that are valuable for diagnosis and organizational governance.

5 Related Work

Commitments have been extensively employed for modeling processes. However, in contrast to our work, most of the previous work has considered only practical commitments. El Menshawy et al. [7] propose the CTLC+ logic for verifying commitments. Their logic handles practical commitments, and includes modalities for commitment creation and fulfillment (or violation). In contrast, our approach handles both practical and dialectical commitments, and considers the

entire lifecycle of commitments not just their creation and fulfillment (or violation). Specifically, CTLC+ cannot handle scenarios in which the debtor cancels its commitment, or the creditor releases the debtor from the commitment.

Winikoff [27] states that agent interactions designed by focusing on messages restrict agent autonomy by limiting their interaction flexibility. He proposes a commitment-based approach for modeling agent interactions. We agree with Winikoff and employ commitments for modeling processes. However, unlike Winikoff, in addition to practical commitments, we consider dialectical commitments as a first class abstraction to model the guarantees made by the participants (agents). Further, we show how agents' interactions can be verified with respect to their commitments.

Singh [21] presents a combined logic for practical and dialectical commitments. He formulates postulates that capture reasoning patterns for commitments. Our work goes beyond Singh's work in proposing an operationalization of dialectical commitments via a new lifecycle, and showing how to employ CTL to formally verify agent interactions. Additionally, we propose novel reasoning patterns incorporating practical and dialectical commitments.

McBurney and Parsons [13], and Krabbe and Walton [11] describe an argumentation-based representation for agent dialogs (interactions), and formal dialectical systems in argumentation, respectively, that include a notion of commitments. However, their approach violates the autonomy of the participants. For example, a question by one agent may "impose a commitment on the second to provide a response" (p. 266). In contrast, we treat a commitment being created autonomously by its debtor. In addition, we provide a formalization of commitments that supports verification.

Some work on architecture for collaboration is relevant even though it does not incorporate commitments. Narendra et al. [15] propose an architecture framework for modeling cross-enterprise collaborations that consists of three layers: strategy, operational, and service layers. The strategy layer specifies the goals and business rules; the operational layer specifies the services; and the service layer specifies the service implementations. Narendra et al.'s framework lacks adequate modeling of the relationships among the participants. It will be interesting to incorporate commitments (practical and dialectical) to capture the relationships among the participants at the strategy layer.

Liptchinsky et al. [12] propose an approach for modeling dynamic collaboration processes that employs a network of collaborative documents and a social network of collaborators. The notion of relations is a fundamental element in Liptchinsky et al.'s modeling approach. It will be interesting to incorporate commitments to model the relations. Commitments provide a rigorous way to capture the relations among the actors such as an actor (or a group of actors) committing to performing certain action or an actor (or a group of actors) making a claim.

Hofreiter et al. [9] present the UMM methodology for modeling global choreographies, that is, interactions among organizations. UMM seeks to specify a choreography at a high level, independently of the underlying implementation technology. However, UMM lacks well-defined abstractions for capturing the relationships underlying the collaborations. Commitments can provide an abstract

and technology independent way of specifying relationships in UMM’s business domain and requirements views.

We agree with Grando et al. [8] regarding the benefits of high-level abstractions for specifying medical processes. However, unlike our approach, Grando et al. take a centralized viewpoint that violates the autonomy of the participants by mandating their goals. Further, since Grando et al.’s approach ignores the social commitments between the participants, it misses specifying the participants’ responsibilities to each other in the modeled process.

Müller et al. [14] describe the importance of interoperability in healthcare but focus on data interoperability, i.e., with respect to message formats. We incorporate considerations of interactions and thus enable specifying and verifying interoperability in general. For example, a radiologist is interoperable with a hospital not only because they agree on the formats of messages they exchange but because they agree on the commitments involved in those messages.

6 Discussion and Future Work

To model sociotechnical systems, such as service engagements, involves modeling the relevant normative relationships or norms properly [22]. Although we consider commitments as the only norm type in this paper, we give first-class status to dialectical commitments, which are a crucial element of secure collaboration. The main new idea of our approach is highlighting the social nature of dialectical commitments. This idea would readily apply to other norm types. We enhance an existing commitment-based process modeling and verification method [25] to incorporate dialectical commitments and organizational context. In healthcare settings, dialectical commitments enable precisely identifying the accountable party behind a diagnosis.

We incorporate our proposed method into a verification approach and tool based on NuSMV. Our representation enables stating important properties of models in high-level terms to capture stakeholder requirements. Our tool can identify potential errors in models, thereby leading to the design of correct STSs.

In future research, we will address some limitations of this work. In particular, on the theoretical side, we will investigate how dialectical commitments relate to other norm types in STSS from the standpoint of foundations of representing, verifying, and achieving secure collaboration in open settings. On the practical side, we will develop and empirically evaluate an enhanced modeling methodology incorporating dialectical commitments as well as a verification method that incorporates an enhanced notion of time to support better representation and verification of STSs. We will also study how commitments relate to existing business process modeling standards such as BPEL [3].

Acknowledgments. Thanks to the anonymous reviewers for helpful comments and to the US Department of Defense for partial support through a Science of Security Lablet grant.

References

1. ASPE. The importance of radiology and pathology communication in the diagnosis and staging of cancer: mammography as a case study, November 2010. Office of the Assistant Secretary for Planning and Evaluation, U.S. Department of Health and Human Services. <http://aspe.hhs.gov/sp/reports/2010/PathRad/index.shtml>
2. Baldoni, M., Baroglio, C., Chopra, A.K., Singh, M.P.: Composing and verifying commitment-based multiagent protocols. In: Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI), pp. 10–17, Buenos Aires, July 2015
3. BPEL. Web services business process execution language, version 2.0, July 2007. <http://docs.oasis-open.org/wsbpel/2.0/>
4. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specification. *ACM Trans. Program. Lang. Syst.* **8**(2), 244–263 (1986)
5. Desai, N., Chopra, A.K., Singh, M.P.: Amoeba: a methodology for modeling and evolving cross-organizational business processes. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* **19**(2), 6:1–6:45 (2009)
6. El-Menshawy, M., Bentahar, J., Dssouli, R.: Modeling and verifying business interactions via commitments and dialogue actions. In: Jędrzejowicz, P., Nguyen, N.T., Howlet, R.J., Jain, L.C. (eds.) *KES-AMSTA 2010, Part II*. LNCS, vol. 6071, pp. 11–21. Springer, Heidelberg (2010)
7. El Menshawy, M., Bentahar, J., El Kholy, W., Dssouli, R.: Reducing model checking commitments for agent communication to model checking ARCTL and GCTL*. *Auton. Agents Multi-Agent Syst.* **27**(3), 375–418 (2013)
8. Grando, M.A., Peleg, M., Glasspool, D.: A goal-oriented framework for specifying clinical guidelines and handling medical errors. *J. Biomed. Inf.* **43**(2), 287–299 (2010)
9. Hofreiter, B., Huemer, C., Liegl, P., Schuster, R., Zapletal, M.: UN/CEFACT’s modeling methodology (UMM): a UML profile for B2B e-commerce. In: Proceedings of the 2nd International Workshop on Best Practices of UML (ER), pp. 19–31 (2006)
10. Kalia, A.K., Telang, P.R., Singh, M.P.: Protos: a cross-organizational business modeling tool (demonstration). In: Proceedings of the 11th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), IFAAMAS, pp. 1489–1490, Valencia, Spain, June 2012
11. Krabbe, E.C.W., Walton, D.: Formal dialectical systems and their uses in the study of argumentation. In: Feteris, E.T., Garssen, B., Francisca Snoeck Henkemans, A. (eds.) *Keeping in Touch with Pragma-Dialectics and Computation*, vol.163, pp. 245–263. Benjamin (2011)
12. Liptchinsky, V., Khazankin, R., Schulte, S., Satzger, B., Truong, H.-L., Dustdar, S.: On modeling context-aware social collaboration processes. *Inf. Syst.* **43**, 66–82 (2014)
13. McBurney, P., Parsons, S.: Dialogue games for agent argumentation. In: Simari, G., Rahwan, I. (eds.) *Argumentation in Artificial Intelligence*, pp. 261–280. Springer, USA (2009)
14. Müller, H., Schumacher, M., Godel, D., Khaled Omar, A., Mooser, F., Ding, S.: Medicoordination: a practical approach to interoperability in the Swiss health system. In: *Proceedings of the Medical Informatics in a United and Healthy Europe (MIE)*, vol. 150, pp. 210–214. IOS Press (2009)

15. Narendra, N.C., Lê, L.-S., Ghose, A., Sivakumar, G.: Towards an architectural framework for service-oriented enterprises. In: Ghose, A., Zhu, H., Yu, Q., Delis, A., Sheng, Q.Z., Perrin, O., Wang, J., Wang, Y. (eds.) ICSOC 2012. LNCS, vol. 7759, pp. 215–227. Springer, Heidelberg (2013)
16. Norman, T.J., Carbogim, D.V., Krabbe, E.C.W., Walton, D.N.: Argument and multi-agent systems. In: Reed, C., Norman, T.J. (eds.) *Argumentation Machines: New Frontiers in Argument and Computation*, Volume 9 of *Argumentation Library*, Chapter 2, pp. 15–54. Kluwer (2004)
17. NuSMV. A new symbolic model checker (2012). <http://nusmv.fbk.eu>
18. Object Management Group, Framingham, Massachusetts. UML 2.0 Superstructure Specification, October 2004
19. Paja, E., Giorgini, P., Paul, S., Meland, P.H.: Security requirements engineering for secure business processes. In: Niedrite, L., Strazdina, R., Wangler, B. (eds.) *BIR Workshops 2011*. LNBIP, vol. 106, pp. 77–89. Springer, Heidelberg (2012)
20. Robinson, W.N., Puro, S.: Specifying and monitoring interactions and commitments in open business processes. *IEEE Softw.* **26**(2), 72–79 (2009)
21. Singh, M.P.: Semantical considerations on dialectical and practical commitments. In: *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)*, pp. 176–181. AAAI Press, Chicago, July 2008
22. Singh, M.P.: Norms as a basis for governing sociotechnical systems. *ACM Trans. Intel. Syst. Technol. (TIST)* **5**(1), 21:1–21:23 (2013)
23. Singh, M.P.: Cybersecurity as an application domain for multiagent systems. In: *Proceedings of the 14th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), IFAAMAS*, pp. 1207–1212. Blue Sky Ideas Track, Istanbul, May 2015
24. Telang, P.R., Kalia, A.K., Singh, M.P.: Engineering service engagements via commitments. *IEEE Internet Comput.* **18**, 1–8 (2014)
25. Telang, P.R., Singh, M.P.: Specifying and verifying cross-organizational business models: an agent-oriented approach. *IEEE Trans. Serv. Comput.* **5**(3), 305–318 (2012). Appendix pp. 1–5
26. Verdicchio, M., Colombetti, M.: Commitments for agent-based supply chain management. *SIGecom Exchan.* **3**(1), 13–23 (2002)
27. Winikoff, M.: Designing commitment-based agent interactions. In: *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pp. 363–370 (2006)

Positron: Composing Commitment-Based Protocols

Scott N. Gerard¹, Pankaj R. Telang², Anup K. Kalia³(✉),
and Munindar P. Singh³

¹ IBM, Research Triangle Park, Durham, NC 27709, USA
sgerard@us.ibm.com

² Cisco, Research Triangle Park, Durham, NC 27709, USA
ptelang@cisco.com

³ NC State University, Raleigh, NC 27695, USA
{akkalia,singh}@ncsu.edu

Abstract. We understand a sociotechnical system (STS) as a micro-society in which social entities interact about and via technical entities. A protocol specifies an STS by describing how its members collaborate by giving meaning to their interactions. We restrict ourselves to protocols that specify messages between roles in terms of how they create and affect commitments among the roles. A key idea of our approach, Positron, is that a protocol specifies the *accountability* of one role to another in addition to the *requirements* from each role. Specifically, Positron incorporates role accountability and role requirements as two integral aspects of protocol composition. In this way, it seeks to promote collaboration in STSs through natural requirements elicitation; flexibility enactment; and compliance and validation (ascribing accountability for each requirement to a specific role). Positron maps composite protocols to the representations of a well-known model checker as a way to verify protocols to assist in their correct formulation. We evaluate Positron by demonstrating it on real-life protocols.

Keywords: Commitments · Commitment protocols · Agent communication · Communication protocols · Protocol composition · Verification of multiagent systems · Model checking

1 Introduction

We study sociotechnical systems (STSS) wherein autonomous parties interact about and through technical entities [29]. STSs arise in a variety of collaborative settings, including cross-organizational service engagements. A protocol specifies an STS in abstract terms by describing two or more roles and the messages those roles may exchange along with meanings of those messages [7]. Protocols arise commonly in business, e.g., RosettaNet [25], and healthcare, e.g., HL7 [16]. By bringing forth collaboration requirements, protocols separate implementations from interactions, thereby promoting the flexibility of autonomous collaborators, such as is needed in sociotechnical systems.

Existing protocol approaches [16,25], however, standardize the message formats and operational constraints, but not the meanings of those messages, which are left informally stated. The past several years have seen the development of approaches that apply *commitments* [26] to specify the meanings of the messages in a protocol [3,34], where protocol designers define meanings. The commitment-based approaches help deal with the autonomy and heterogeneity of participants and promote flexibility in interactions. The benefits of commitments for modeling service engagements are well established. For example, Telang and Singh [30] demonstrated an error in a published RosettaNet guideline when modeled using commitments. Therefore, for brevity, we do not review the extensive literature on commitments here. Singh [28] provides a conceptual summary.

Challenge: Composition. Composition is a key construct in software engineering as a way to promote reuse and modularity. Existing approaches for protocols, whether operational or commitment-based, do not adequately support their composition, because they incorporate internal details or lack a formal semantics of interactions. Existing approaches, e.g., HL7 [16] and RosettaNet [25], provide atomic two-party protocols with the intent for them to be composed but do not support the composition as such and do not provide a construct by which two or more protocols could be composed. Thus the separation of interaction and implementation fails above the level of the atomic (predefined) protocols. Section 5 reviews the literature in detail. Suffice it to state here that previous relevant research falls into these categories: (a) composition but no commitments [22,27]; (b) commitments but no composition [14]; and (c) composition and commitments. The last subcategory can be further refined as (c1) purely abstract description without a specification language or tools [19]; (c2) composition of commitment-based protocols based on axioms [9,11] or regulative constraints [4] but without producing a composite protocol for further reuse and composition; and (c3) our present approach to composition of commitment-based protocols based on role responsibilities and accountabilities.

Motivating Research Questions and Contributions. How can we (1) formalize accountability, a crucial element of secure collaboration, as a basis for formal modeling and verification of STSs and (2) support composition of protocols to specify STSs? We address this question by restricting ourselves to composition and verification of commitment protocols, deferring representing other normative relationships [29] and their mapping to agent decision-making to future work.

Specifically, we propose *Positron*, a language and verification approach that supports composing commitment-based protocols and formally reasoning about them to verify desired properties. Positron (a) introduces *role requirements*, which capture a role’s motivation, and *role accountability*, which captures the commitments a role makes to other creditor roles (that benefit from those commitments) as elements of a composite protocol specification; (b) shows how to recursively expand nested constituent protocols; (c) supports a methodology for composing commitment protocols; and (d) provides a decision procedure and mechanical verification of protocols with respect to role requirements, role

accountabilities, and enactments. Positron compiles protocol specifications into MCMAS [17] models and checks protocols against temporal logic formulas. It then employs the MCMAS model checker to verify if the composite protocol satisfies those temporal formulas. The Positron verifier builds upon the Proton [14], verifier for commitment protocol refinement, expanding it to tackle protocol composition.

2 Background and Motivation

We write $C_{\{\text{debtors}\},\{\text{creditors}\}}(\textit{antecedent},\textit{consequent})$ [14] to denote a commitment *from* the specified debtors *to* the specified creditors that if the antecedent begins to hold, the debtors will bring about the consequent. The antecedent and consequent are Boolean expressions. For example, $C_{PH,\{CC,Re\}}(\textit{deliverReq} \wedge \textit{approval},\textit{deliverCar})$ denotes that POLICYHOLDER (PH) commits to CALLCENTER (CC) and REPAIRER (Re) that he will deliver his car to REPAIRER whenever requested and the repair request is approved.

When the antecedent becomes true, the commitment is *detached*, and the debtors become *unconditionally* committed to the creditors. When the consequent becomes true, the commitment is *discharged*. Debtors *should* discharge their detached commitments. However, debtors are autonomous and may violate a commitment, for simplicity, by canceling it. The only computational requirement for commitments is: each detached commitment *must* eventually be discharged (satisfied, delegate, assigned, or released) or canceled. A commitment imposes no ordering constraint between the antecedent and consequent, although in specific settings there may be a practical constraint.

Running Example: AGFIL. The following real-life case involves automobile insurance claims processing for AGF Irish Life Holding (AGFIL) [5], as Fig. 1 summarizes. This case involves four parties plus POLICYHOLDER and ADJUSTER (not shown). AGFIL underwrites automobile insurance policies and covers losses incurred by policy holders. Europ Assist (EA) provides a 24-h help-line service for receiving claims. Approved REPAIRERS provide repair services. Lee Consulting Services (Lee) coordinates with AGFIL, repairers, and adjusters to handle a claim.

Figure 1 describes the workflows of each participant along with how they relate to one another. Notwithstanding that one could adopt a standard process notation, the main point is that such a description tightly couples the inner workings of the participants. Conventional protocol approaches [16, 25, 32] deemphasize the inner workings and capture the interactions between the participants via a formal notation in terms of constraints on the ordering of the messages exchanged between the participants.

In contrast, a commitment protocol emphasizes the social state of an interaction, expressed in terms of commitments. A commitment protocol describes the roles involved, the messages they exchange, and any preconditions and effects of the messages on the social state. An agent adopts a role and enacts the specified

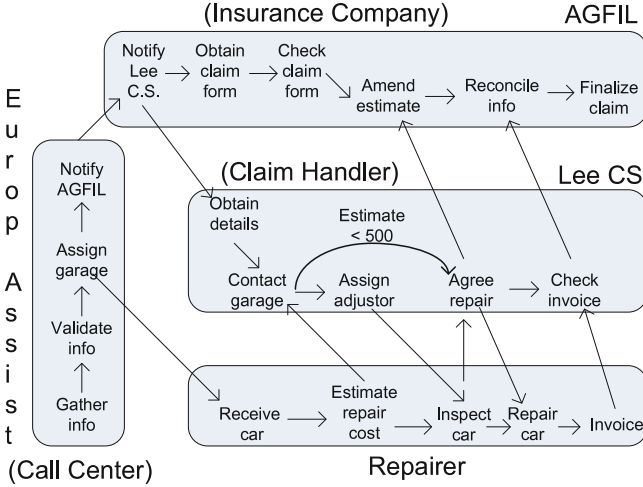


Fig. 1. Traditional process model of cross-organizational insurance claim processing [5].

protocol by autonomously choosing (in accordance with its internal policies) how to interact.

3 Technical Approach

Positron provides a formal language in which to express composite protocols based on existing constituent protocols. Positron is a Java application that reads protocols described in the Positron language (examples shown in Listings 1 and 2), flattens any hierarchically nested protocols, and generates input to the MCMAS model checker.

Recall that Proton [14] provides a language for capturing roles, propositions, commitments, and messages. Positron augments the Proton language by adding constructs to define a composite protocol using a set of parameterized constituent protocols and defines a protocol composition methodology.

Further, while it accepts and verifies any CTL expression, Positron introduces five constructs for common verification patterns when composing protocols: Function *Req* for role requirements, *coupling commitments* for role accountabilities, and three *path expressions* for good and bad enactments.

Definition 1. A Positron protocol is a six-tuple of \mathcal{P}^R , a set of role names; \mathcal{P}^P , a set of role-qualified propositions; \mathcal{P}^C , a set of commitments; \mathcal{P}^M , a set of guarded messages; \mathcal{P}^F , a set of CTL expressions to be verified; and \mathcal{P}^U , a set of use (include) statements.

For brevity, we omit the Positron grammar in favor of examples. Listing 1 specifies AGFIL’s Claim Handling protocol as roles (insured and claims handler), propositions, commitments, and messages (with their guards and effects on the propositions).

Listing 1. Claim constituent protocol

```

1: protocol Claims (role In, role CH, prop notify, prop estimate, prop invoice, prop pay) {
2:   role In; CH;
3:   prop notify; estimate; invoice; pay;
4:   commitment
5:     cc = CIn,CH(estimate and invoice, pay);
6:     notify.cc1 = CCH,In(notify, estimate);
7:     pay.c1 = CIn,CH(invoice, pay);
8:     pay.c2 = CCH,In(pay, invoice);
9:   message
10:    In → CH : [true] notify_requestMsg means {notify};
11:    CH → In : [notify.isSet()] notify_responseMsg means {estimate};
12:    In → CH : [true] pay_do1 means {pay};
13:    CH → In : [true] pay_do2 means {invoice};
14: }
```

Listing 2 describes the AGFIL composite protocol. In the statement beginning on Line 4, this protocol uses the Claims protocol in Listing 1, mapping roles and propositions in protocol AGFIL to equivalent versions in protocol Claims. We omit the other constituent protocols for space. It turns out that the AGFIL protocol has no additional messages since all its messages derive from its constituents. The AGFIL protocol includes formulas describing the correctness requirements.

Listing 2. Positron specification for AGFIL protocol (partial)

```

1: protocol AGFIL {
2:   role PH; In; CC; CH; Re; Ad;
3:   prop accident; deliver; repair; paid; ...
4:   use IN-CH : Claims(agfil.Claims,
5:     role In = IN, role CH=CH,
6:     prop notify = true, prop estimate=reportCH,
7:     prop pay = payCH, prop invoice=repairIn);
8:   ...
9:   commitment
10:    CC1 : CCC,{PH,Re}(deliverReq, notifyRE);
11:    PH1 : CPH,{CC,Re}(deliverReq ∧ approval, deliver);
12:    ...
13:   formula
14:    AG(paid ∧ accident → AF(repair ∨ anyCancel));
15:    Req(IN, coverage ∧ premium ∧ accident, repair);
16:    AG(¬(repair ∧ ¬inspectCH));
17:    ...
18:    EFPPath(accident, deliverReq, payCa, reportCH, reportRe, ..., repair);
19:    ¬EFPPath(repair, accident);
20:    ...
21: }
```

Definition 2. An MCMAS representation is a tuple of \mathcal{M}^{agent} , a set of agent names, including a distinguished agent Env representing the environment; \mathcal{M}^{state} , a set of agent-qualified variable names; \mathcal{M}^{msg} , a set of agent-qualified guarded transition functions; \mathcal{M}^{evol} , a set of agent-qualified evolution expressions; \mathcal{M}^{init} , a set of agent-qualified variable initializations; \mathcal{M}^{eval} , a mapping from propositions to expressions over variables in \mathcal{M}^{state} ; \mathcal{M}^{fair} , a set of fairness expressions; and \mathcal{M}^{ctl} , a set of CTL expressions to be verified.

Protocol Composition. Positron supports nested composition of protocols. A composite protocol P can use (include) a parameterized constituent protocol with a use statement $q : Q(\bar{x} = \bar{p})$ specifying protocol name (q), protocol type (Q), a set of arguments \bar{p} passed by a composite protocol P , and a matching set of parameters \bar{x} accepted by constituent Q . Arguments and parameters are named and have a type of either role or proposition. The argument and parameter sets must contain matching names and types. Positron expands any hierarchical nesting in P to produce a single, flat protocol P' . Expansion gives every element in Q a new, unique name, and replaces each parameter with its corresponding argument. Unique names are constructed by prepending the constituent name q to each element name in Q . Positron supports using multiple copies or instances of the same constituent Q by using distinct names q and q' .

Definition 3. Given a set of arguments \bar{p} , a parameterized constituent protocol type Q accepts a set of parameters \bar{x} , where the sets \bar{p} and \bar{x} agree in both name and type. Define $Q_{\bar{p}}^{\bar{x}}$ as Q in which all elements in Q are given unique names, and every parameter in \bar{x} is replaced with its corresponding argument in \bar{p} .

Expanding a composite P containing a constituent $q : Q(\bar{x} = \bar{p})$ yields the union of P and $Q_{\bar{p}}^{\bar{x}}$, and removing P 's use statement $q : Q(\bar{x} = \bar{p})$.

Definition 4. Given a composite protocol P that uses a constituent protocol $q : Q(\bar{x} = \bar{p})$, where P passes a set of arguments \bar{p} , Q accepts a set of parameters \bar{x} , and the sets \bar{p} and \bar{x} agree in name and type. Then protocol $P' = \text{expand}(P, q : Q(\bar{x} = \bar{p}))$ is the expanded version of P and Q , and is defined as follows, where $x \in \{\text{R}, \text{P}, \text{C}, \text{M}, \text{F}\}$

$$\begin{aligned} \mathcal{P}^x(P') &:= \mathcal{P}^x(P) \cup \mathcal{P}^x(Q_{\bar{p}}^{\bar{x}}) \\ \mathcal{P}^u(P') &:= (\mathcal{P}^u(P) - q) \cup \mathcal{P}^u(Q_{\bar{p}}^{\bar{x}}). \end{aligned}$$

Positron Conversion to MCMAS. The conversion of a Positron protocol, \mathcal{P} , to an MCMAS representation $\mathcal{M} = \text{conv}(\mathcal{P})$ is a two-step process. First, constituent protocol expansion (Protocol Composition) flattens all nested protocols, ensuring \mathcal{P}^u is the empty set. Second, additional conversion functions, $\text{conv}_m^p(\mathcal{P}^x)$, convert each element of a Positron protocol to elements of an MCMAS representation. The final MCMAS representation consolidates these generated elements. The ISPL source input into MCMAS is generated from the above-mentioned MCMAS representation.

Role Requirements. A role requirement reflects a role-desired goal of an agent playing a role in the composite protocol. In Positron, $Req(r, p, q)$ means that role r requires that q will occur whenever p occurs. In AGFIL, one of POLICYHOLDER’s role requirements is: if INSURER offers coverage, I paid the premium, and I have an accident, then my car will be repaired: $Req(\text{PH}, \text{coverage} \wedge \text{premium} \wedge \text{accident}, \text{repair})$

It is incorrect to formalize a role requirement as the CTL specification: $\mathbf{AG}(\text{accident} \rightarrow \mathbf{AF} \text{ repair})$, which ignores commitment violations that disrupt a collaboration: a commitment may fail because its debtor either chooses not to, or is prevented by circumstances from, discharging it. In verifying a role requirement, we cannot assume commitments are never canceled. Rather, we express role r ’s requirement as: if r fulfills all its own commitments and p holds at any state, then on each branch eventually, either q holds or a role other than r canceled one of its commitments. If r ’s requirement fails because r cancels a commitment, that is not a fault of the protocol; it is r ’s fault.

A Positron role requirement maps to an MCMAS CTL expression as

$$Req(r, p, q) := \mathbf{AG}(p \rightarrow \mathbf{AF}(q \vee \bigvee_{r' \neq r} r'.\text{anyCancel}))$$

where $r'.\text{anyCancel}$ is true if and only if role r' cancels any of its commitments.

Enactment Requirements. Although capturing all possible enactments is not feasible for all protocols, designers and other stakeholders often know of specific good and bad enactments. We use these *enactments* for partially verifying a composite protocol as a way to assist designers refine protocol specifications (e.g., its constituent protocols and coupling commitments) or other requirements. An enactment corresponds to a scenario in requirements engineering [13] and yields a unit test. In Positron, the enactment specifications can be “good” (must exist) or “bad” (must not exist).

We use model checking to verify enactments. We introduce three recursive functions to simplify enactment specification. An enactment E is an ordered list of Boolean expressions over states and messages. $\text{head}(E)$ is the first element in list E , and $\text{tail}(E)$ is E without the first element, and \mathbf{EX} , \mathbf{EF} and \mathbf{EU} are CTL operators. Define

$$\begin{aligned} EXPath(E) &:= \begin{cases} \text{head}(E) \wedge \mathbf{EX}(EXPath(\text{tail}(E))) & \text{if } |E| > 1 \\ \mathbf{EX}(P) & \text{if } |E| = 1 \end{cases} \\ EFPPath(E) &:= \begin{cases} \mathbf{EF}(\text{head}(E) \wedge EFPPath(\text{tail}(E))) & \text{if } |E| > 1 \\ \mathbf{EF}(P) & \text{if } |E| = 1 \end{cases} \\ EUPath(r, E) &:= \begin{cases} \mathbf{E}(\neg r \mathbf{U} (\text{head}(E) \wedge EUPath(r, \text{tail}(E)))) & \text{if } |E| > 1 \\ \mathbf{E}(\neg r \mathbf{U} P) & \text{if } |E| = 1 \end{cases} \end{aligned}$$

$EXPath$ specifies a path of states that must appear consecutively. $EXPath$ is often too strong a constraint, since it precludes interleaving of constituent protocols. $EFPPath$ specifies a path of states that must appear in order, but not necessarily consecutively. $EUPath$ specifies a path of states that must appear in

order, and constrains which states can be interleaved in the path. Expression r identifies which states must *not* be interleaved in the path. An $EUPath$ requirement is stronger than $EFPPath$ and weaker than $EXPath$. Two example requirements from AGFIL include $\neg EFPPath(\text{repair}, \text{accident})$ and $EFPPath(\text{accident}, \text{deliverReq}, \text{deliverCar}, \dots, \text{repair})$.

Coupling Commitments. A composite protocol would in general relate its constituent protocols. All roles are jointly accountable for ensuring constituent protocols are properly interrelated. We capture each role’s *role accountabilities* as *coupling commitments*. A coupling commitment’s debtor is the accountable role, and its creditors are (in general) the union of all roles connected by the interrelated constituent protocols, minus the debtor: $C_{\text{accountable role}, \{\text{interrelated roles}\}}(\text{antecedent}, \text{consequent})$.

Consider two coupling commitments from AGFIL. (1) CALLCENTER commits to POLICYHOLDER and REPAIRER that it will notify REPAIRER whenever it receives a request: $C_{CC, \{PH, Re\}}(\text{deliverReq}, \text{notifyRE})$. (2) POLICYHOLDER commits to CALLCENTER and REPAIRER that he will deliver his car to REPAIRER whenever requested and the repair request is approved: $C_{PH, \{CC, Re\}}(\text{deliverReq} \wedge \text{approval}, \text{deliverCar})$.

Verification. Positron reads specifications of the composite and constituent protocols and generates a single MCMAS input file. MCMAS reads the input, builds the appropriate model, and reports whether each CTL formula holds in the model.

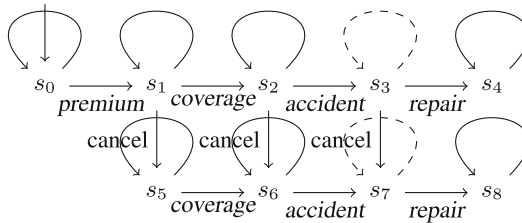


Fig. 2. Selected states and transitions for AGFIL.

Figure 2 shows a portion of the state space Positron generates for verification from AGFIL’s constituent protocols and coupling commitments. The start state is s_0 . Solid lines are valid transitions (messages); dashed lines are transitions that must not occur infinitely often. Since the message guard for *coverage* is *premium*, *coverage* can occur only after *premium*, making $s_1 \dots s_8$ invalid start states. Notice that the top row (*premium*, *coverage*, *accident*, and *repair*) begins a good enactment. Positron can verify the existence of this path using an $EFPPath$ requirement. Further, Positron can ensure that the model is free of specific bad enactments, for example, that s_1 must not be a start state.

Positron generates a model checking fairness constraint for each commitment: a commitment must not remain unconditional and unresolved forever. Dashed

loops are invalid because they violate a commitment fairness constraint. A composite protocol may fail to satisfy role or enactment requirements for different reasons:

- *Fail_{RR}*: If a role requirement (*Req*) formula fails, then coupling commitments are missing; add coupling commitments that require agents to act as appropriate.
- *Fail_G*: If a good enactment formula fails, then either (*Fail_{GG}*) some message guards are too strong; weaken guards to enable additional good transitions. Or (*Fail_{GC}*) some commitment can become detached, but can never resolve; weaken guards to enable transitions that satisfy the commitment’s consequent.
- *Fail_B*: If a bad enactment formula fails, then some message guards are too weak; strengthen guards to disable existing incorrect transitions.

4 Evaluation of Positron Modeling and Tools

We evaluate our contributions by modeling real-life protocols from the insurance, manufacturing, and healthcare domains. Table 1 summarizes our iterative methodology to develop a composite protocol such as in Listing 2.

AGFIL Evaluation. We extend the AGFIL scenario by adding (a) POLICYHOLDER role and accident reporting; (b) ADJUSTER role and the redirection of two messages between CLAIMHANDLER and REPAIRER through ADJUSTER; (c) payments from INSURER to CLAIMHANDLER and REPAIRER; (d) a protocol for premiums and coverage between POLICYHOLDER and INSURER; and (e) REPAIRER returning the car.

We create the AGFIL protocol of Listing 2 as follows. **Roles:** Identify roles: INSURER (IN) for agent AGFIL, CALLCENTER (CC) for EUROP ASSIST, CLAIMHANDLER (CH) for LEE, POLICYHOLDER (PH), REPAIRER (RE), and

Table 1. Inputs and outputs for each step of the methodology.

Step	Name	Inputs	Outputs
1	Roles	Background and requirements	Composite roles
2	Constituent selection	Role relationships and protocol library	Constituent protocols
3	Role requirements	Role’s needs	Role requirements
4	Enactments	Background knowledge of requirements	Good and bad enactments
5	Coupling	Enactments	Coupling commitments
			Composite protocol
6	Positron	All artifacts	Protocol specification
7	Verification	Protocol specification	Model checker results

ADJUSTER (AD). **Constituent Selection:** Identify constituent protocols: *RequestResponse*, *Exchange* (where two roles swap items), *Claims* for IN-CH and *ApprovedWork* for CH-RE. **Role Requirements:** Identify role requirements: POLICYHOLDER requires: (1) if he has coverage, pays his premium, and has an accident, his car is repaired; (2) if he delivers his car to REPAIRER, his car is returned. INSURER requires: if a claim is filed, the claim is finalized. All roles except POLICYHOLDER require payment if they perform their tasks. All these are described as Req functions. **Enactments:** Identify enactments: (a) POLICYHOLDER reports an accident to CALLCENTER (PH-CC); (b) CALLCENTER assigns and notifies REPAIRER to repair the car (CC-RE); (c) CALLCENTER asks POLICYHOLDER to deliver his car to a specific REPAIRER (PH-CC); (d) POLICYHOLDER delivers car to REPAIRER (PH-RE); remaining steps are omitted. Performing repairs before an accident is reported is a bad enactment: (e) car repaired; (f) accident reported. **Coupling:** Identify coupling commitments: (1) Between messages (a) and (b) of the accident-reporting enactment (see previous step), if POLICYHOLDER reports an accident, CALLCENTER assigns and notifies REPAIRER and (2) between messages (c) and (d), if CALLCENTER asks POLICYHOLDER to deliver his car to REPAIRER, he does so. **Positron:** Generate the Positron specification for AGFIL. Listing 2 shows snippets of the Positron specification for AGFIL protocol. Lines 2 and 3 declare roles and propositions. Line 4 instantiates constituent *Claims* named IN-CH. Lines 10 and 11 are two coupling commitments. Line 15 lists one of POLICYHOLDER’s role requirements. Line 16 lists an INSURER requirement as explicit CTL. Line 18 verifies the good, accident-reporting enactment that must exist in the composite, and Line 19 verifies a bad enactment that must not exist. Listing 1 is the Claims protocol used as a constituent protocol. From all previous artifacts, generate the MCMAS input files. **Verification:** Run MCMAS model checker.

Quote To Cash Evaluation. Quote To Cash (QTC) is an important business process that supports manufacturing supply chains [24]. **Roles:** Identify roles: CUSTOMER, RESELLER, DISTRIBUTOR, SELLER, SHIPPER1, and SHIPPER2. **Constituent Selection:** Identify constituent protocols: CUSTOMER orders goods and services from RESELLER using constituent protocol *CommercialTran* (*ComTran*), RESELLER fulfills the order by *Outsourcing* to DISTRIBUTOR, DISTRIBUTOR orders good from SELLER using *CommercialTran*, SELLER arranges shipping with SHIPPER2, and DISTRIBUTOR arranges shipping with SHIPPER1, using additional instances of *Outsourcing*, and SELLER provides a customer support contract to CUSTOMER though *StandingService*. **Role Requirements:** Identify role requirements; if CUSTOMER pays, he receives goods and services, if RESELLER pays DISTRIBUTOR, he receives shipment, and for CUSTOMER whenever a role performs its task, it gets paid. **Enactments:** Identify two enactments for CUSTOMER placing an order and ending with fulfillment: one if DISTRIBUTOR has goods in stock, one if it restocks from SELLER. Fulfilling an order before it is verified is a bad enactment. **Coupling:** Identify coupling commitments: (1) CUSTOMER couples CU-RE and CU-RE-DI: if CUSTOMER receives a shipment, he pays RESELLER, (2) RESELLER couples CU-RE and CU-RE-DI:

whenever RESELLER receives an order, he orders from DISTRIBUTOR. **Positron:** Generate Positron specification for QTC. **Verification:** Run MCMAS model checker.

Healthcare (ASPE) Evaluation. We consider the healthcare process for breast cancer diagnosis, as described by an HHS committee [2]. The resulting ASPE protocol contains five roles (for convenience, we associate feminine pronouns with PATIENT, RADIOLOGIST, and REGISTRAR and masculine pronouns with PHYSICIAN and PATHOLOGIST.) The process begins when PATIENT visits a primary care physician (PHYSICIAN), who detects a suspicious mass in her breast. He sends PATIENT to RADIOLOGIST for a mammography. If RADIOLOGIST notices suspicious calcifications, she sends a report to PHYSICIAN recommending a biopsy. PHYSICIAN requests the RADIOLOGIST to perform a biopsy, who collects a tissue specimen from PATIENT and sends it to a PATHOLOGIST. PATHOLOGIST analyzes the specimen, and performs ancillary studies. If necessary, PATHOLOGIST and RADIOLOGIST confer to reconcile their results and produce a consensus report. PHYSICIAN reviews the integrated report with PATIENT to create a treatment plan. PATHOLOGIST forwards his report to REGISTRAR who adds PATIENT to a state-wide cancer registry. There are only two coupling commitments in ASPE. PHYSICIAN has no coupling commitments because it is his choice whether PATIENT needs mammogram and biopsy exams from RADIOLOGIST.

Real-Life Results:

Positron and MCMAS were run on all three, hierarchical examples with the statistics and timings for the final, corrected protocols shown in Table 2. Positron processing was quick at less than 4s, and total processing of AGFIL and ASPE were also quick at less than 8s. QTC, with a 1000 times larger state space and the most CTL formulas, required 21 minutes.

Table 2. Statistics. (M is 10^6 , G is 10^9 , s is seconds.)

Composite metric	AGFIL	QTC	ASPE
Constituent instances	11	6	12
Roles	6	6	5
Propositions	22	37	18
Commitments (total)	24	43	12
Coupling commitments	9	21	2
Messages	22	55	20
CTL formulas (total)	9	17	14
Role requirements	8	13	7
Enactment requirements	1	4	7
Positron statements	94	164	81
State space size	120 M	381 G	1.47 M
Positron processing time	1.98 s	3.16 s	1.68 s
MCMAS processing time	4.29 s	1274 s	5.78 s
Total time	6.27 s	1278 s	7.46 s

Model Verification: Resolving verification errors is a challenging task, requiring careful consideration of the interactions between the generated MCMAS model and CTL statements.

Positron helped identify and fix several verification failures. Good enactment failures ($Fail_{GG}$ and $Fail_{GC}$) were generally easier to fix, since they only require

that some path exist. One or more requirements or coupling commitments were slightly weakened to allow additional branches at appropriate points. Bad enactment failures ($Fail_B$) were harder to fix as they required the impossibility of a particular enactment. One or more role requirements or coupling commitments were added. Hand checking was insufficient to ensure the changes would eliminate all bad paths; only reruns of Positron and MCMAS could confidently verify the changes.

Model Validation: Although identifying requirements was easy, some initial specifications were incorrect because preconditions were missed. For example, POLICYHOLDER’s role requirement initially failed ($Fail_{RR}$) because coupling commitment among CALLCENTER, POLICYHOLDER, and REPAIRER did not include *approval*; REPAIRER will not repair a car just because it is delivered to him. And, an accident is insufficient to get POLICYHOLDER’s car is repaired; POLICYHOLDER must also have a policy and pay the premium. Positron generates model checking fairness conditions to ensure all unconditional commitments eventually resolve. Initially, one of the AGFIL’s good enactment mysteriously failed because, even though the model allowed all the transitions, the good enactment had unresolvable commitments (invalid by commitment fairness conditions). We corrected the model so all commitments could resolve.

5 Discussion: Literature and Future Work

Positron gains an advantage over both traditional process modeling and existing (operational) protocol approaches by focusing on high-level relationships realized as constituent protocols, and by focusing on commitments rather than control flow. Because role accountabilities are stated as commitments, if a requirement fails, we can trace the failure back to a specific failing role.

Composite protocols provide a formal means to capture how constituent protocols may be composed to realize an STS specification. Because these protocols capture meanings as commitments (generalizable to norms), yet have a formal semantics that maps to sound enactments, they can provide a natural approach to support secure policy-governed collaboration in ways that are not visible to low-level, operational approaches.

Literature. Table 3 compares Positron with other work. Some papers propose a protocol specification language, and some propose an accompanying protocol specification methodology. Some papers address single protocols in isolation; some address common patterns within protocols; some address the composition of multiple protocols to create new composite protocols. Of those papers that address verification, some address sociotechnical requirements; some address verification properties between two protocols or models (such as protocol refinement); some address protocol-wide properties; some verify properties that must hold between the constituents of a composite protocol; some formulate role-specific properties; some formulate good or bad enactment properties; and some address other verification topics not addressed above.

Table 3. Approach comparison. Column abbreviations and citations are Po = Positron; Pr = Proton; DA, DO, Dv and DM = Desai et al.; T = Telang and Singh; Y = Yolum; Mi = Miller and McBurney; G = Günay et al.; C = Cheong and Winikoff, Mc = McGinnis and Robertson, L = Lomuscio et al., B = Baldoni et al. Check marks show the significant topics addressed by each paper. The cell contents of the verification rows indicate whether the paper discusses (D) or mechanizes (M) verification of known good or bad paths.

Significant topics	Po	Pr	DA	DO	Dv	DM	T	Y	Mi	G	C	Mc	L	B
Verification topics		[14]	[9]	[10]	[8]	[11]	[31]	[33]	[21]	[15]	[6]	[20]	[18]	[4]
Protocol specification	✓	✓	✓	✓	–	✓	✓	✓	✓	✓	✓	✓	–	✓
Methodology	✓	–	✓	✓	–	–	✓	–	–	✓	✓	–	–	–
Single protocol	–	✓	✓	✓	✓	–	–	✓	✓	✓	–	✓	✓	✓
Protocol patterns	–	–	–	–	–	–	✓	–	–	–	–	–	–	–
Protocol-to-protocol verification	–	✓	–	–	–	–	?	–	–	–	–	?	?	–
Protocol composition	✓	–	✓	✓	✓	✓	–	–	✓	–	✓	–	✓	✓
Requirements verification														
Sociotechnical	M	–	–	–	M	–	M	–	–	D	–	–	M	M
Protocol-to-protocol	–	M	–	–	–	–	M	–	–	–	–	–	M	–
Protocol	M	–	–	–	M	–	–	M	M	D	D	–	–	M
Inter-constituent	M	–	–	–	–	M	–	–	–	–	–	–	–	–
Role-specific	M	–	–	–	–	–	–	–	–	D	–	–	M	–
Enactments	M	–	–	–	–	–	M	–	–	–	D	–	M	M
Other	M	M	–	–	M	M	M	M	M	D	–	–	M	M

Desai et al. [9] propose OWL-P [10] and MAD-P [11] for specifying and verifying commitment protocols and their compositions. They employ axioms to specify a composition. These approaches suffer from a key drawback: axiom violations are not assigned to any particular role. In contrast, Positron employs coupling commitments with clear role accountability for the effects of one constituent protocol on others. Further, Amoeba is purely manual, whereas Positron incorporates mechanical verification. Adopting Amoeba’s event ordering idea would add flexibility to our approach, but more granular parameterizations of constituents provides the same functionality.

Telang and Singh [31] (T&S) describe a methodology for modeling STSs that captures the commitments to be created among the parties by melding selected collaboration patterns. In contrast, a protocol in Positron additionally specifies the messages and guards, and the protocols are first-class entities that retain their identity in the composite protocol, yielding improved modularity and modifiability. Most significantly, T&S’s approach verifies if one implementation is sound with respect to the model. In contrast, Positron verifies if the model itself is sound.

Yolum [33] proposes generic correctness properties of commitment protocols for design-time verification, but does not address composite protocols. She considers generic properties, whereas we consider role-specific STS requirements. It would be interesting to formulate Yolum’s generic correctness properties in Positron.

Miller and McBurney [21] (M&M) propose the *RASA* language based on propositional dynamic logic (PDL) to specify and compose protocols. *RASA*'s preconditions, actions and postconditions correspond to Positron's guards, messages and meanings. Positron additionally incorporates role requirements, coupling commitments, and good and bad enactment paths, making Positron practically viable. These are important in naturally describing STS protocols, as we demonstrated above. Whereas M&M describe a custom reasoner, we rely on standard CTL semantics as realized in MCMAS.

Günay et al. [15] treat protocols as sets of commitments and propose automatically generating such sets from an agent's beliefs, goals, and capabilities. In contrast, we offer a semiautomatic approach where a tool helps designers compose existing protocols. Automatic generation is attractive but may not be feasible for complex settings, although a hybrid approach of developing atomic protocols mechanically and composite protocols with human assistance might be viable.

Cheong and Winikoff [6] describe the Hermes system for goal-oriented interaction. They focus on interaction-level goals, whereas we focus on role-level requirements and commitments. Their action sequence diagrams capture only good enactments.

McGinnis and Robertson [20] propose an approach in which an agent sends a protocol specification to other agents at runtime, as a way to accomplish dynamic, runtime, protocol adaptation. They remark that their approach lacks a way to prevent agents from making an undesirable change to a protocol. If their protocols were augmented with commitments, Positron could help address this gap. For example, an agent may not remove a message from a protocol that brings about the consequent of a detached commitment. While they describe rules for dynamically changing protocols, they do not address formal verification of interaction properties.

Lomuscio et al. [18] semiautomatically compile and verify contract-regulated service compositions with compliance expressed in temporal-epistemic logic using MCMAS (which we adopt). A crucial difference is that Lomuscio et al. consider service compositions; we consider protocol compositions. Since a protocol has a distributed footprint, protocol compositions are inherently more subtle than service compositions. A potential benefit from adopting MCMAS is that it supports more expressive logics such as Alternating-Time Temporal Logic (ATL) [1], which could help capture subtle correctness criteria for protocols.

We propose a methodology and use role responsibilities and role accountabilities using expressions in CTL. Others describe the complementary issue of *temporal pattern languages* which assist users to correctly capture their high level requirements as temporal expressions. Dwyer et al. [12] describe a pattern language for temporal expressions in CTL, LTL and other formalisms. Baldoni et al. [4] compose protocols using regulative specifications as LTL constraints.

BPMN 2.0 [23] is a standard notation for business process modeling. BPMN addresses both orchestration and choreography; in taking a multiagent approach, Positron focuses on choreography. Our protocols are similar to BPMN

Conversation objects. Both protocols and Conversations can involve two or more roles (Participants), and both can have simple or complex message sequencing relationships. But, BPMN does not specify how to describe the relationships between messages in a complex Conversation, except through internal orchestration; our complex protocols fully specify message sequencing relationships using external guard statements. Both support arbitrary nesting. We formally verify our compositions using temporal logic; model verification is explicitly out-of-scope in the BPMN specification. Positron’s primary building blocks are protocols and commitments. BPMN has no counterpart to our coupling commitments, which are key to our interrelating constituent protocols and formal verification.

Threats, Limitations, and Future Directions. One limitation of Positron is that it does not handle varieties of accountability besides commitments [29] and does not show how to evaluate agent goals and decision making with respect to protocols [15]. Importantly, we have not established that practitioners employing Positron can obtain the benefits in abstraction, reusability, and correctness the motivate it.

These threats and limitations lead to useful future directions. At the theoretical level, treating the goals of the participants is natural. At the practical level, generating enactments via tooling would be valuable. At the empirical level, evaluating the effectiveness of Positron (the approach and the tool) with professional developers on collaboration in STSs would be necessary to promote adoption by industry. To this end, we have developed a graphical notation for protocol composition called *composite protocol diagrams* (CPDs). CPDs seek to succinctly visualize the essence of a composite protocol both to analysts and technical designers, who collaborate in its construction. We defer an empirical evaluation of CPDs and competing notations as a way to determine if the high-level abstractions of Positron can help analysts and designers combat complexity and communicate more effectively with each other.

Acknowledgments. Thanks to the anonymous reviewers for helpful comments and to the US Department of Defense for partial support through a Science of Security Lablet grant.

References

1. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM **49**(5), 672–713 (2002)
2. ASPE: the importance of radiology and pathology communication in the diagnosis and staging of cancer (November 2010). Assistant Secretary for Planning and Evaluation, U.S. Department of Health and Human Services. <http://aspe.hhs.gov/sp/reports/2010/PathRad/index.shtml>
3. Baldoni, M., Baroglio, C., Marengo, E.: Behavior-oriented commitment-based protocols. In: Proceedings of the 19th European Conference on Artificial Intelligence (ECAI), pp. 137–142, August 2010

4. Baldoni, M., Baroglio, C., Marengo, E., Patti, V., Capuzzimati, F.: Engineering commitment-based business protocols with the 2CL methodology. *J. Auton. Agents Multi-Agent Syst. (JAAMAS)* **28**(4), 519–557 (2014)
5. Browne, S., Kellett, M.: Insurance (motor damage claims) scenario. Document D1.a, CrossFlow Consortium (1999)
6. Cheong, C., Winikoff, M.P.: Hermes: designing flexible and robust agent interactions. In: Dignum, V. (ed.) *Handbook of Research on Multi-Agent Systems*, Chap. 5. IGI Global (2009)
7. Chopra, A.K., Dalpiaz, F., Aydemir, F.B., Giorgini, P., Mylopoulos, J., Singh, M.P.: Protos: foundations for engineering innovative sociotechnical systems. In: *Proceedings of the 18th IEEE International Requirements Engineering Conference (RE)*, pp. 53–62, August 2014
8. Desai, N., Cheng, Z., Chopra, A.K., Singh, M.P.: Toward verification of commitment protocols and their compositions. In: *Proceedings of the 6th International Conference on Autonomous Agents Multiagent Systems*, pp. 33:1–33:3. ACM (2007)
9. Desai, N., Chopra, A.K., Singh, M.P.: Amoeba: a methodology for modeling and evolving cross-organizational business processes. *ACM Trans. Soft. Eng. Methodol. (TOSEM)* **19**(2), 6:1–6:45 (2009)
10. Desai, N., Mallya, A.U., Chopra, A.K., Singh, M.P.: Interaction protocols as design abstractions for business processes. *IEEE Trans. Soft. Eng.* **31**(12), 1015–1027 (2005)
11. Desai, N., Singh, M.P.: A modular action description language for protocol composition. In: *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI)*, pp. 962–967, July 2007
12. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: *Proceedings of the International Conference on Software Engineering (ICSE)*, pp. 411–420, May 1999
13. Filippidou, D.: Designing with scenarios: a critical review of current research and practice. *Requirements Eng.* **2**(1), 1–22 (1998)
14. Gerard, S.N., Singh, M.P.: Formalizing and verifying protocol refinements. *ACM Trans. Intell. Syst. Technol. (TIST)* **4**(2), 21:1–21:27 (2013)
15. Günay, A., Winikoff, M., Yolum, P.: Commitment protocol generation. In: *Proceedings of the 10th AAMAS Workshop Declaration Agent Languages and Technologies (DALT)*, pp. 51–66, June 2012
16. HL7: Health Level Seven (2007). <http://www.hl7.org>
17. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: a model checker for the verification of multi-agent systems. In: Bouajjani, A., Maler, O. (eds.) *CAV 2009*. LNCS, vol. 5643, pp. 682–688. Springer, Heidelberg (2009)
18. Lomuscio, A., Qu, H., Solanki, M.: Towards verifying contract regulated service composition. *J. Auton. Agents Multi-Agent Syst. (JAAMAS)* **24**(3), 345–373 (2012)
19. Mallya, A.U., Singh, M.P.: An algebra for commitment protocols. *J. Auton. Agents Multi-Agent Syst. (JAAMAS)* **14**(2), 143–163 (2007)
20. McGinnis, J., Robertson, D.: Dynamic and distributed interaction protocols. In: Kudenko, D., Kazakov, D., Alonso, E. (eds.) *AAMAS 2004*. LNCS (LNAI), vol. 3394, pp. 167–184. Springer, Heidelberg (2005)
21. Miller, T., McBurney, P.: Propositional dynamic logic for reasoning about first-class agent interaction protocols. *Comput. Intel.* **27**(3), 422–457 (2011)

22. Miller, T., McGinnis, J.: Amongst first-class protocols. In: Artikis, A., O'Hare, G.M.P., Stathis, K., Vouros, G.A. (eds.) ESAW 2007. LNCS (LNAI), vol. 4995, pp. 208–223. Springer, Heidelberg (2008)
23. Object Management Group: Business Process Model and Notation (2011). <http://bpmn.org/>
24. Oracle: Automating the Quote-to-Cash process, June 2009. <http://www.oracle.com/us/industries/045546.pdf>
25. RosettaNet: Home page (2009). <http://www.rosettanel.org>
26. Singh, M.P.: An ontology for commitments in multiagent systems: toward a unification of normative concepts. *Artif. Intel. Law* **7**(1), 97–113 (1999)
27. Singh, M.P.: Information-driven interaction-oriented programming. In: Proceedings of the 10th International Conference on Autonomous Agents MultiAgent Systems (AAMAS), pp. 491–498, May 2011
28. Singh, M.P.: Commitments in multiagent systems. In: Paglieri, F., et al. (eds.) *The Goals of Cognition*, Chap. 32, pp. 613–638. College Publications (2012)
29. Singh, M.P.: Norms as a basis for governing sociotechnical systems. *ACM Trans. Intel. Syst. Technol. (TIST)* **5**(1), 21:1–21:23 (2013)
30. Telang, P.R., Singh, M.P.: Abstracting and applying business modeling patterns from RosettaNet. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) *ICSOC 2010*. LNCS, vol. 6470, pp. 426–440. Springer, Heidelberg (2010)
31. Telang, P.R., Singh, M.P.: Specifying and verifying cross-organizational business models: an agent-oriented approach. *IEEE Trans. Serv. Comput.* **5**(3), 305–318 (2012)
32. WS-CDL: Web Services Choreography Description Language, Version 1.0, November 2005. <http://www.w3.org/TR/ws-cdl-10/>
33. Yolum, P.: Design time analysis of multiagent protocols. *Data Knowl. Eng. J.* **63**(1), 137–154 (2007)
34. Yolum, I., Singh, M.P.: Commitment machines. In: Meyer, J.-J.C., Tambe, M. (eds.) *ATAL 2001*. LNCS (LNAI), vol. 2333, pp. 235–247. Springer, Heidelberg (2002)

Analysis of Timing Constraints in Heterogeneous Middleware Interactions

Ajay Kattapur¹(✉), Nikolaos Georgantas², Georgios Bouloukakis²,
and Valérie Issarny²

¹ PERC, TCS Innovation Labs, Mumbai, India
ajay.kattapur@tcs.com

² MiMove Team, Inria Paris-Rocquencourt, Paris, France
{nikolaos.georgantas,georgios.bouloukakis,valerie.issarny}@inria.fr

Abstract. With the emergence of *Future Internet* applications that connect web services, sensor-actuator networks and service feeds, scalability and heterogeneity support of interaction paradigms are of critical importance. Heterogeneous interactions can be abstractly represented by *client-service*, *publish-subscribe* and *tuple space* middleware connectors that are interconnected via bridging mechanisms providing interoperability among the services. In this paper, we make use of the *eXtensible Service Bus* (XSB), proposed in the *CHOReOS* project as the connector enabling interoperability among heterogeneous choreography participants. XSB models transactions among peers through generic **post** and **get** operations that represent peer behavior with varying time/space coupling. Nevertheless, the heterogeneous **lease** and **timeout** constraints of these operations severely affect latency and success rates of transactions. By precisely studying the related timing thresholds using *timed automata* models, we verify conditions for successful transactions with XSB connectors. Furthermore, we statistically analyze through simulations, the effect of varying **lease** and **timeout** periods to ensure higher probabilities of successful transactions. Simulation experiments are compared with experiments run on the XSB implementation testbed to evaluate the accuracy of results. This work can provide application developers with precise design time information when setting these timing thresholds in order to ensure accurate runtime behavior.

Keywords: Heterogeneous services · Middleware interoperability · Interaction paradigms · Timed automata · UPPAAL · Statistical analysis

1 Introduction

Service Oriented Architectures (SOA) allow heterogeneous components to interact via standard interfaces and by employing standard protocols. Choreographies [4] of such components allow large scale integration of devices (exposed as

V. Issarny—This work has been partially supported by the European Union’s Horizon 2020 Research and Innovation Programme H2020/2014–2020 under grant agreement number 644178 (project CHOReVOLUTION, <http://www.chorevolution.eu>).

services) via SOA. However, these principally use the *client-service* interaction paradigm, as for instance, with RESTful services [21]. With the advent of paradigms such as the *Internet of Things* [13] that involve not only conventional services but also sensor-actuator networks and data feeds, additional middleware level abstractions are needed to ensure interoperability.

In particular, heterogeneous platforms, such as REST [21] supporting *client-service* interactions, *publish-subscribe* based Java Messaging Service [20], or JavaSpaces [11] offering a shared *tuple space*, can be made interoperable through middleware protocol converters [15]. In this paper, we use the *eXtensible Service Bus* (XSB) proposed by the *CHOReOS* project¹ [9, 12] for dealing with heterogeneous choreographies at the middleware level. XSB prescribes a connector that abstracts and unifies the three aforementioned interaction paradigms: *client-service* (CS), *publish-subscribe* (PS) and *tuple space* (TS). Furthermore, XSB is implemented as a common bus protocol that enables interoperability among services employing heterogeneous interactions following one of these paradigms.

While our previous work [16] studies the effect of heterogeneous choreographies on multi-dimensional end-to-end QoS properties, we now analyze heterogeneous middleware interactions with specific emphasis on timing behavior. We propose a timing model that can represent a system relying on not only any of the CS, PS, TS paradigms, but also any interconnection between them. This model can be used to compare between paradigms, select among them, tune the timing parameters of the overlying application, and also do the previous when interconnection is involved. Our model captures data availability and validity in time with the `lease` parameter, as well as intermittent availability of the data recipients with the `timeout` parameter. Hence, this model allows us to study, in a unified manner, time coupling and decoupling among interacting peers.

We examine the conditions for successful transactions with timed automata [2], and verify reachability and safety properties by employing the UPPAAL [6] model-checker. This analysis provides us with formal conditions for successful XSB transactions and their reliance on the `lease` and `timeout` parameters as well as on the stochastic behavior of interacting peers. We further perform statistical analysis through simulation of transactions over multiple runs, and study the success rate and latency trade-off with varying `lease` and `timeout` periods. Simulation outputs are compared with experiments run on the XSB testbed with respect to the accuracy of predicted results. By analyzing the related timing thresholds, we enable designers to leverage the `lease` and `timeout` periods effectively in order to obtain maximal transaction success rates. Moreover, designers can evaluate the impact of interconnecting heterogeneous systems having different timing behaviors, or the impact of replacing a middleware paradigm by another.

The rest of the paper is organized as follows. An overview of heterogeneous interaction paradigms and XSB is provided in Sect. 2. The model for timing analysis of XSB transactions is introduced in Sect. 3. This is further refined with timed automata models and verification of properties in Sect. 4. The results of our analysis through simulation experiments are presented in Sect. 5, which includes

¹ <http://www.choreos.eu/bin/view/Main/>.

comparison with experiments on the XSB implementation. This is followed by related work and conclusions in Sects. 6 and 7, respectively.

2 Interconnecting Heterogeneous Interaction Paradigms

To deal with heterogeneous service choreographies of the *Future Internet*, we make use of the modeling solution proposed in the *CHOReOS* [9] project. While typical service choreographies utilize pure client-service interactions between participants, Future Internet applications require inclusion of service feeds (via publish-subscribe) and sensor-actuator networks (via shared tuple spaces). We briefly review salient properties of these interaction paradigms:

Table 1. XSB connector API.

Primitives	Arguments
<code>post</code>	<code>mainscope, subscope, data, lease</code>
<code>get</code>	<code>↑mainscope, ↑subscope, ↑data, timeout</code>

- *Client-Service (CS)* is a commonly used paradigm for web services. A client (**source**) communicates directly with a server (**destination**) either by direct messaging (one-way **send**) or by a remote procedure call (RPC, two-way) through an **operation**. Both synchronous and asynchronous reception of messages (**receive**) are possible at the receiving entity (within a **timeout** period). CS represents tight *space coupling*, with the client and service having knowledge of each other. There is also tight *time coupling*, with service availability being crucial for successful message passing.
- *Publish-Subscribe (PS)* is a commonly used paradigm for content broadcasting/feeds. Peers interact using an intermediate **broker** service; publishers produce (**publish**) events characterized by a specific topic (**filter**) to the broker; subscribers subscribe their interest for specific topics to the broker; and the broker matches received events with subscriptions and delivers a copy of each event to an interested subscriber (**retrieve**) until a **lease** period. PS allows *space decoupling*, as the subscribers need not know each other. Additionally, *time decoupling* is possible, with the disconnected subscribers receiving updates synchronously or asynchronously when reconnected to the broker.
- *Tuple Space (TS)* is commonly used for shared data with multiple read/write users. Peers interact with a *tuple space* (**tspace**) and have write (**out**), read and tuple removal (**take**) access to the commonly shared data. Further, peers are able to choose a **template** to select the tuples they procure from the tuple space. TS enables both space and time decoupling between interacting peers.

We employ the *eXtensible Service Bus (XSB) connector*², which ensures interoperability across the above interaction paradigms. XSB extends the conventional ESB system integration paradigm [12]. The XSB API is depicted in

² <http://xsb.inria.fr/>.

Table 2. APIs of interaction paradigms mapped to XSB primitives.

Interaction	Native primitives	XSB primitives
CS	send (destination, operation, message)	post (destination, operation, message, 0)
	receive (↑source, ↑operation, ↑message, timeout)	get (↑source, ↑operation, ↑message, timeout)
PS	publish (broker, filter, event, lease)	post (broker, filter, event, lease)
	retrieve (↑broker, ↑filter, ↑event, timeout)	get (↑broker, ↑filter, ↑event, timeout)
TS	out (tspace, template, tuple, lease)	post (tspace, template, tuple, lease)
	take (↑tspace, ↑template, ↑tuple, timeout)	get (↑tspace, ↑template, ↑tuple, timeout)
	read (↑tspace, ↑template, ↑tuple, timeout)	get (↑tspace, ↑template, ↑tuple, timeout)

Table 1, where we only refer to *one-way* interactions; *two-way* interactions are built by combining two of the former. It employs primitives such as **post** and **get** to abstract CS (**send**, **receive**), PS (**publish**, **retrieve**), and TS (**out**, **take/read**) interactions. The **data** argument can represent a CS message, PS event or TS tuple. The **mainscope** and **subscope** arguments are used to unify space coupling (addressing mechanisms) across CS, PS and TS. We employ the ↑ symbol in Table 1 to denote a return argument of a primitive. XSB primitives and arguments can be mapped one-to-one to typical primitives and arguments of CS, PS and TS as shown in Table 2.

We exploit the semantics of the **lease** and **timeout** parameters in each interaction paradigm as follows: **lease** refers to emitted messages, events or tuples, and characterizes both data availability in time, e.g., thanks to storing by a broker, and data validity, e.g., for data that become obsolete as part of a data feed. Hence, **lease** equals to zero in the client-service paradigm, as shown in Table 2. **timeout** characterizes the interval during which a receiving peer is connected and available. During this active period, the peer can receive one or more sent messages, events or tuples, either synchronously or asynchronously. Between active periods, the peer is disconnected, e.g., for energy-saving or other application-related reason. In the next section, we study in further detail the effect of these parameters on successful interactions.

3 Timing Analysis of Interactions

In this section, we examine the timing thresholds for **timeout/lease** periods and their effects on successful data passing in choreography interactions, where **data** is our generic representation of messages, events and tuples. We examine, in a unified manner, both time (de)coupled and synchronous/asynchronous data

passing. This analysis is critical for inter-operating heterogeneous distributed systems, where the designer has to reconcile varying system timing behaviors.

In order to enrich choreography interactions with timing constraints, we focus on *one-way transactions* over a client-service, publish-subscribe, or tuple space connection, abstractly represented by the XSB connector: a transaction represents an end-to-end interaction enabling posting and getting of data. We examine latency increments δ for such transactions. Our analysis considers in particular the “steady state” behavior of publish-subscribe and tuple space interactions. For PS, this means that subscribers are already subscribed and do not unsubscribe during the study period. For TS, this means that peers accessing the tuple space properly coordinate for preventing early removal of data by one of the peers before all interested peers have accessed this data.

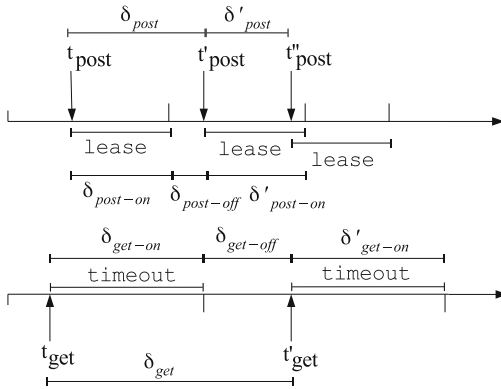


Fig. 1. Analysis of *post* and *get* δ increments.

In an XSB transaction, a *poster* entity posts data with a validity period **lease**; this data can be procured using **get** within the **timeout** period at the *getter* side. Figure 1 depicts a XSB transaction as a correlation in time between a **post** operation and a **get** operation. The **post** and **get** operations are independent and have individual time-stamps. We assume that application entities (undertaking the poster and getter roles) enforce their semantics independently (no coordination). The **post** operation is initiated at t_{post} . A timer is started also at t_{post} , constraining the data availability to the **lease** period $\delta_{\text{post-on}}$. Note that the $\delta_{\text{post-on}}$ period may be set to 0, as in the case of CS messages. The period when the **lease** period elapses and the next **post** operation is yet to begin is denoted as $\delta_{\text{post-off}}$.

Similarly at the getter side, the **get** operation is initiated at t_{get} , together with a timer controlling the active period limited by the **timeout** interval, denoted as $\delta_{\text{get-on}}$. If **get** returns within the **timeout** period with valid data (not exceeding the **lease**), then the transaction is successful. We consider this instance also as the end of the **post** operation. **post** operations are initiated

repeatedly, with an interval rate δ_{post} (set as a random valued variable) between two successive **post** operations. Similarly, **get** operations are initiated repeatedly, with a random valued interval equal to δ_{get} between the start of two successive $\delta_{\text{get-on}}$ periods; the interval between **timeout** and the next t_{get} qualifies the disconnection period of receivers ($\delta_{\text{get-off}}$). While **lease** and **timeout** are in general set by application/middleware designers, inter-arrival delays δ_{post} and δ_{get} are stochastic random variables dependent on multiple factors such as concurrent number of peers, network availability, user (dis)connections and so on.

Note that this model allows concurrent **post** messages; buffers of active receiving entities (including the broker and tuple space) are assumed to be infinite. The data processing, transmission and queueing (due to processing and transmission of preceding data) times inside the transaction are assumed to be negligible (or of the same order in the CS case of $\delta_{\text{post-on}} \approx 0$) compared to durations of $\delta_{\text{post-on}}$ and $\delta_{\text{get-on}}$ periods. In particular regarding queueing, we assume that we have no heavy load effects. This means that: all posts arriving during an active period are immediately served; all posts arriving during an inactive period are immediately served at the next $\delta_{\text{get-on}}$ period, unless they have expired before. This corresponds to a G/G/ ∞ / ∞ queueing model, where there are an infinite number of on-demand servers, hence there is no queueing. We assume that the general distribution characterizing service times incorporates the disconnections of receivers. Extending this model with actual queueing is part of our ongoing unfinished work.

Successful transactions depend on either of the disjunctive conditions:

$$t_{\text{get}} < t_{\text{post}} < t_{\text{get}} + \text{timeout} \quad (1)$$

$$t_{\text{post}} < t_{\text{get}} < t_{\text{post}} + \text{lease} \quad (2)$$

meaning that a successful transaction occurs as long as a **post** and a **get** operation overlap in time. Otherwise, there is no overlapping in time between the two operations: only one of them takes place, and goes up to its maximum duration, i.e., **lease** for **post** and **timeout** for **get**. Precisely:

1. If **get** occurs first, and then **post** occurs before **timeout**: the transaction is *successful*. Else, **timeout** is reached, and the **get** operation yields no transaction.
2. If **post** occur first, and then **get** occurs before **lease**: the transaction is *successful*. Else, **lease** is reached, and the transaction is a *failure*.

The above analysis of XSB transactions not only can represent the individual CS, PS, TS interactions, but also any heterogeneous interconnection between them, e.g., a PS publisher interacting with a TS reader. Interconnection is performed through the XSB bus, i.e., an ESB-style middleware implementing the XSB connector. We assume that the effect of the XSB bus on the timings of the end-to-end interactions is negligible.

4 Timed Automata Model

In this section, we build a timed automata model that represents the typical behavior of the XSB connector and of application components using this connector for performing the timed interactions described in the previous section. By relying on the expressive power of timed automata, we are able not only to model the timing conditions of such interactions, but also to introduce basic stochastic semantics for the poster and getter behavior. Using the UPPAAL model checker, we provide and verify essential properties of our timed automata model, including formal conditions for successful XSB transactions.

4.1 Timed Automata Model of XSB

A timed automaton [2] is essentially a finite automaton extended with real-valued clock variables. These variables model the logical clocks in the system, which are initialized with zero when the system is started, and then increase synchronously at the same rate. Clock constraints are used to restrict the behavior of the automaton. A transition represented by an edge can be taken only when the clock values satisfy the *guard* labeled on the edge. Clocks may be reset to zero when a transition is taken. Clock constraints are also used as *invariants* at locations represented by vertices: they must be satisfied at all times the location is reached or maintained.

In order to study XSB interactions with timed automata, we make use of UPPAAL [6]. UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata. In such networks, automata synchronize via *binary synchronization channels*. For instance, with a channel declared as `chan c`, a transition of an automaton labeled with `c!` (sending action) synchronizes with the transition of another automaton labeled with `c?` (receiving action). UPPAAL makes use of computation tree logic (CTL) [10] to specify and verify temporal logic properties. We employ the *committed location* qualifier (marked with a ‘C’) for some of the locations. In UPPAAL, time is not allowed to pass when the system is in a committed location; additionally, outgoing transitions from a committed location have absolute priority over normal transitions. The *urgent location* qualifier (marked with a ‘U’) is also used: time is not allowed to pass when the system is in an urgent location, either (without the priority clause of committed locations, though).

We represent XSB transactions with the connector roles *XSB poster*, *XSB getter*, and with the corresponding *XSB glue*. The two roles model the behavior expected from application components employing the connector, while the glue represents the internal logic of the connector coordinating the two roles. We detail in the following the modeling of these components.

Figure 2 shows the *poster* behavior. Typically, a poster entity repeatedly emits a `post!` message to the *glue* without receiving any feedback about the end (successful or not) of the `post` operation. We have enhanced (and at the same time constrained) the poster’s behavior with a number of features. The committed locations `post_event` (`post!` sent to the glue) and `post_end_event` (`post_end?`

received from the glue) have been introduced to detect the corresponding events. Upon these events, the automaton oscillates between the `post_on` and `post_off` locations, which correspond to the $\delta_{\text{post-on}}$ and $\delta_{\text{post-off}}$ intervals presented in Fig. 1. `delta_post` is a clock that controls the δ_{post} interval between two successive `post` operations. `delta_post` is reset upon a new `post` operation and set to `lease` at the end of this operation (note that the `post_init` location and its outgoing transition serve initializing `delta_post` at the beginning of the poster's execution – this unifies verification also for the very first `post` operation). The invariant condition `delta_post` \leq `max_delta_post` (where `max_delta_post` is a constant) at the `post_off` location ensures that a new `post` operation will be initiated before the identified boundary. This setup results in at most one `post` operation active at a time. This `post` remains active ($\delta_{\text{post-on}}$ interval) for `lease` time (and then it expires) or less than `lease` time (in case of successful transaction). In both cases, we set `delta_post` to `lease` at the end of the `post` operation (this enables verification, since we can not capture absolute times in UPPAAL). Hence, the immediately following $\delta_{\text{post-off}}$ interval will last a stochastic time uniformly distributed in the interval $[\text{lease}, \text{max_delta_post}]$. With regard to the timing model of Sect. 3, we opted here for restraining concurrency of `post` operations for simplifying the architecture of the glue. The present model (poster, getter and glue) can be compared to one of the infinite on-demand servers of the G/G/ ∞ / ∞ model of Sect. 3. Nevertheless, this model is sufficient for verifying Conditions (1) and (2) for successful XSB transactions. These conditions relate any `post` operation with an overlapping `get` operation; possible concurrency of `post` operations has no effect on this. Moreover, we will see that these conditions are independent of the probability distributions characterizing the poster and getter's stochastic behavior.

Figure 3 shows the *getter* behavior. Typically, a getter entity repeatedly emits a `get!` message to the glue, with at most one `get` operation active at a time. The duration of the `get` operation is controlled by the getter with a local `timeout`; upon the `timeout`, a `get_end!` message is sent to the glue. Before reaching the `timeout`, multiple data items (posted by posters) may be delivered to the getter by the glue, each with a `get_return?` message. We have enhanced the getter's behavior with similar features as for the poster. Hence, we capture the events and time intervals presented in Fig. 1 with the `get_event`, `get_end_event`, `get_on`, `get_off` locations, as well as with the `delta_get` clock and the invariant conditions `delta_get` \leq `timeout` (at `get_on`) and `delta_get` \leq `max_delta_get` (at `get_off`). This setup results in a succession of $\delta_{\text{get-on}}$ and $\delta_{\text{get-off}}$ intervals, with the former lasting `timeout` time and the latter lasting a stochastic time uniformly distributed in the interval $[\text{timeout}, \text{max_delta_get}]$. We have additionally introduced the committed location `no_trans`, which, together with the Boolean variable `get_ret`, helps detecting whether the whole `timeout` period elapsed with no transaction performed or at least one data item was received.

The *glue* automaton is shown in Fig. 4. It determines the synchronization of the incoming `post?` and `get?` operations. A successful synchronization between such operations leads to a successful transaction, which is represented in the

automaton by the `trans_succ` location. Note that the timing constraints specified in Sect. 3 regarding the lifetime of posted data have been applied here with the additional clock `delta_post_on` employed to guard transitions dependent on the `lease` period. Two ways for reaching the `trans_succ` location are considered:

- If the `get?` operation occurs from the initial location (leading to location `glue_get`), a consequent `post?` operation results in a `get_return!` message and eventually the successful transaction location `trans_succ` (Eq. 1). At the same time, the poster is notified of the end of the `post` operation with `post_end!`. Note that we employ the *urgent location* qualifier for `glue_get_post`; thus, the

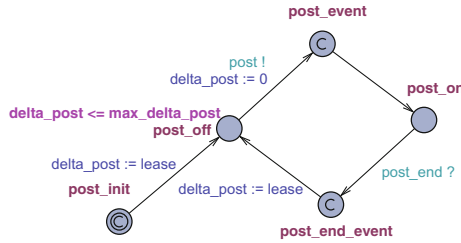


Fig. 2. XSB poster automaton.

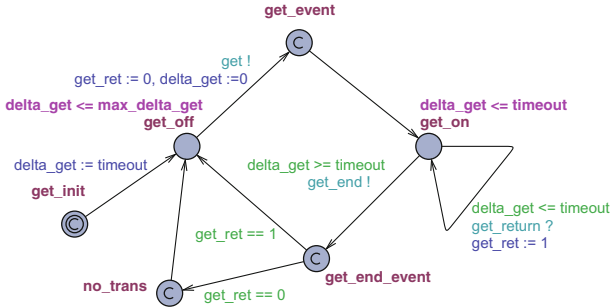


Fig. 3. XSB getter automaton.

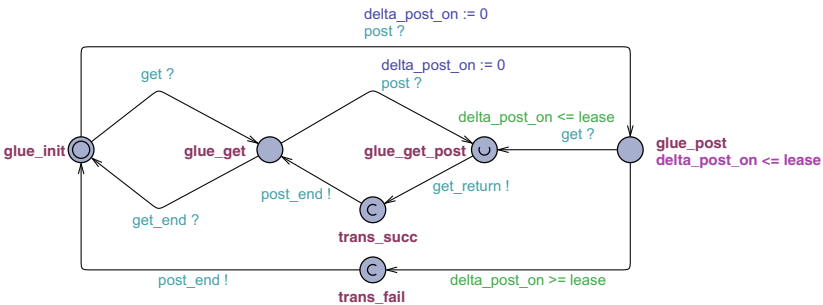


Fig. 4. XSB glue automaton.

glue completes instantly the successful transaction and is ready for a new one. At the `glue_get` location, if the `get_end?` message is received from the getter automaton (suggesting `delta_get >= timeout`), the glue is reset to the initial location `glue_init`.

- If the `post?` operation occurs initially (leading to location `glue_post`), a `get?` operation before the constraint `delta_post_on <= lease` results again in a successful transaction (Eq. 2). Exceeding the `lease` period without any `get?` results in location `trans_fail`, and the automaton returns to its initial location `glue_init`, notifying at the same time the poster with `post_end!`. This is done without any delay, thanks to the invariant `delta_post_on <= lease` at the `glue_post` location.

4.2 Verification of Properties

We verify reachability and safety properties of the combined automata *XSB poster*, *XSB getter* and *XSB glue*, by using the model checker of UPPAAL. A reachability property, specified in UPPAAL as $E \langle \rangle \varphi$, expresses that, starting at the initial state, a path exists such that the condition φ is eventually satisfied along that path. A safety property, specified in UPPAAL as $A [] \varphi$, expresses that the condition φ invariantly holds in all reachable states.

Poster Automaton. We verify a set of reachability and safety properties that characterize the timings of the poster’s stochastic behavior.

$$A [] \text{poster.post_event imply } \text{delta_post} == 0 \quad (3)$$

$$A [] \text{poster.post_on imply } \text{delta_post} \leq \text{lease} \quad (4)$$

$$A [] \text{poster.post_off imply } (\text{delta_post} \geq \text{lease and} \\ \text{delta_post} \leq \text{max_delta_post}) \quad (5)$$

$$E \langle \rangle \text{poster.post_end_event and } \text{delta_post} < \text{lease} \quad (6)$$

Equation 3 states that `post` events occur at time 0 captured by the `delta_post` clock. Equations 4 and 6 together state that $[0, \text{lease}]$ is the maximum interval in which a `post` operation is active, nevertheless, the operation can end before `lease` is reached. Equation 5 states that $[\text{lease}, \text{max_delta_post}]$ is the maximum interval in which there is no active `post` operation. This confirms the fact that we artificially “advance time” to `lease` at the end of the `post` operation.

Getter Automaton. We verify similar properties that characterize the timings of the getter’s stochastic behavior.

$$A [] \text{getter.get_event imply } \text{delta_get} == 0 \quad (7)$$

$$A [] \text{getter.get_on imply } \text{delta_get} \leq \text{timeout} \quad (8)$$

$$A [] \text{getter.get_off imply } (\text{delta_get} \geq \text{timeout and} \\ \text{delta_get} \leq \text{max_delta_get}) \quad (9)$$

$$A[] \text{ getter.get_end_event imply delta_get==timeout} \quad (10)$$

Hence, Eq. 7 states that `get` events occur at time 0 captured by the `delta_get` clock. Equations 8 and 10 together state that a `get` operation precisely and invariantly terminates at the end of the $[0, \text{timeout}]$ interval. Equation 9 states that $[\text{timeout}, \text{max_delta_get}]$ is the maximum interval in which there is no active `get` operation.

Glue Automaton. Finally, we verify conditions for successful transactions using the glue automaton.

$$A[] \text{ glue.trans_succ imply (poster.post_on and getter.get_on} \\ \text{and (delta_post==0 or delta_get==0))} \quad (11)$$

In addition to the reachability property ($E \langle \rangle \text{ glue.trans_succ}$), we verify the safety property in Eq. 11. According to this, a successful transaction event implies that while a `post` operation is active a `get` event occurs, or while a `get` operation is active a `post` event occurs.

$$A[] \text{ glue.trans_fail imply (poster.post_on and getter.get_off} \\ \text{and delta_post==lease and delta_get-timeout>=lease)} \quad (12)$$

In addition to the reachability property ($E \langle \rangle \text{ glue.trans_fail}$), we verify the safety property in Eq. 12. A failed transaction event means that `lease` is reached for an active `post` operation and no `get` operation is active. Additionally, the ongoing inactive `get` interval entirely includes the terminating active `post` interval. With regard to the stochastic `post` and `get` processes of our specific setting, we explicitly checked that if the condition $\text{max_delta_get-timeout} \geq \text{lease}$ does not hold for the given values of the included constants, then the reachability property $E \langle \rangle \text{ glue.trans_fail}$ is indeed not satisfied.

$$A[] \text{ getter.no_trans imply (getter.get_on and poster.post_off} \\ \text{and delta_get==timeout and delta_post-lease>=timeout)} \quad (13)$$

In addition to the reachability property ($E \langle \rangle \text{ getter.no_trans}$), we verify the safety property in Eq. 13. Symmetrically to Eq. 12, a no-transaction event implies that `timeout` is reached for an active `get` operation and no `post` operation is active. Additionally, the ongoing inactive `post` interval entirely includes the terminating active `get` interval. Similarly to Eq. 12, we check that if this safety property is not satisfied, then the state `getter.no_trans` is indeed not reachable.

Checking Eqs. 11, 12, 13, successful transactions are determined by the durations and relative positions in time of the $\delta_{\text{post-on}}$, $\delta_{\text{post-off}}$, $\delta_{\text{get-on}}$ and $\delta_{\text{get-off}}$ intervals. These depend on the deterministic parameter constants `lease`, `timeout` and on the stochastic parameters δ_{post} and δ_{get} . Nevertheless, Eqs. 11, 12, 13 are expressed in a general way, independently of the specific `post` and `get` processes. Hence, the analysis results of this section provide us with general formal conditions for successful XSB transactions and their reliance on observable and potentially tunable system and environment parameters. Using these results, we perform experiments to quantify the effect of varying these parameters for successful transactions in the next section.

5 Results: Analysis of Timing Thresholds

In this section, we provide results of simulations of XSB transactions with varied `timeout` and `lease` periods. We demonstrate that varying these periods has a significant effect on the rate of successful transactions. In case of choreographies, the trade-off involved between success rates and latency (depending on `timeout/lease` periods) is also evaluated.

5.1 Transaction Success Rates

In order to test the effect of varying `lease` and `timeout` periods on transaction success rates, we perform simulations over the timing analysis model described in Sect. 3. *Poisson* arrival rates are assumed for subsequent t_{post} instances (hence, δ_{post} follows the corresponding exponential distribution). The data is valid for a deterministic `lease` period and then discarded. Similarly, there are *exponential* intervals between subsequent t_{get} periods (δ_{get} follows this distribution). The getter entity is active for a deterministic `timeout` period and can disconnect for random valued intervals. Applying the timing model in Sect. 3, the simulation enables concurrent posts with no-queueing. As the arrivals follow a *Poisson* process, this simulates an M/G/ ∞ / ∞ queueing model.

The simulations done in *Scilab*³ analyze the effect of varying `lease` and `timeout` periods on XSB transactions. We set δ_{post} between subsequent `post` messages to have a mean of 10s. The `get` messages are simulated with varying exponential active periods (δ_{get}). This procedure was run for 10,000 t_{get} periods to collect transaction statistics, by applying the formal conditions of Sect. 4.

The rates of successful transactions are shown in Fig. 5 for various values of `lease`, `timeout` and δ_{get} periods. As expected, increasing `timeout` periods for individual `lease` values improves the success rate. However, notice that the success rate is severely bounded by `lease` periods. For time/space coupled CS interactions, where the `lease` period is very low (0s), the success rate, even at

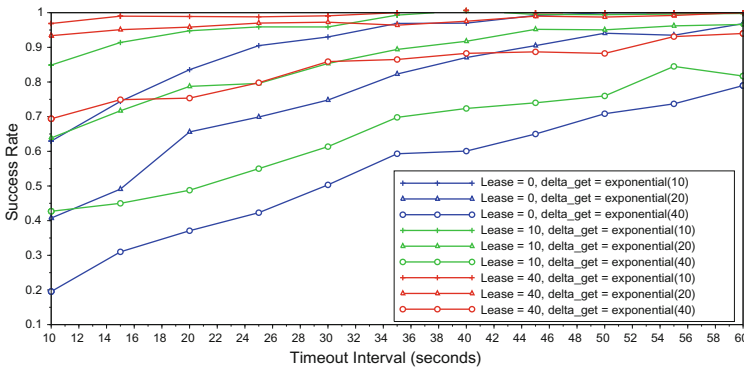


Fig. 5. Transaction success rates with varying `timeout` and `lease` periods.

³ <http://www.scilab.org>.

higher `timeout` intervals, remains bound at around 70 % for δ_{get} with mean 40 s. Reducing `get` disconnection intervals (by properly setting δ_{get} and `timeout`) produces a significant improvement in the success rate, especially for the CS case. For the other interaction paradigms (PS/TS), where the `lease` period can be varied: a higher `lease` period combined with higher `timeout` or lower δ_{get} intervals would guarantee better success rates.

5.2 Latency vs. Success Rate

In order to study the trade-off between end-to-end latency and transaction success rate, we present cumulative latency distributions for transactions in Fig. 6. Note that we assume that all posts arriving during an active `get` period are immediately served; all posts arriving during an inactive `get` period are immediately served at the next active period, unless they have expired before. All failed transactions are pegged to the value: `lease`.

We set $\delta_{\text{post}} = \text{Poisson}(10)$ s and $\delta_{\text{get}} = \text{Exponential}(20)$ s for all simulated cases. From Fig. 6, lower `lease` periods produce markedly improved latency. For instance, with `lease` = 10 s, `timeout` = 20 s, all transactions complete within 10 s. Comparing this to Fig. 5, the success rate with these settings is 78 %. Changing to `lease` = 40 s, `timeout` = 20 s, we get a success rate of 95 %, but with increased latency. So, with higher levels of `lease` periods (typically PS/TS), we notice high success rates, but also higher latency. While individual success rates and latency values depend also on the network/middleware efficiency, our analysis provides general guidelines for setting the `lease` and `timeout` periods to ensure successful transactions.

We provide in the following an illustrative use case, where our fine-grained timing analysis can be employed to properly configure a concrete application. In a transport information management system based on both authoritative and mobile crowd-sourced information from multiple heterogeneous sources, `posts` carrying events of interest for the average user arrive with a mean rate of 1 event every 10 min. To guarantee the freshness of provided information, notifications are maintained by the system for a `lease` period of 10 min. We assume that

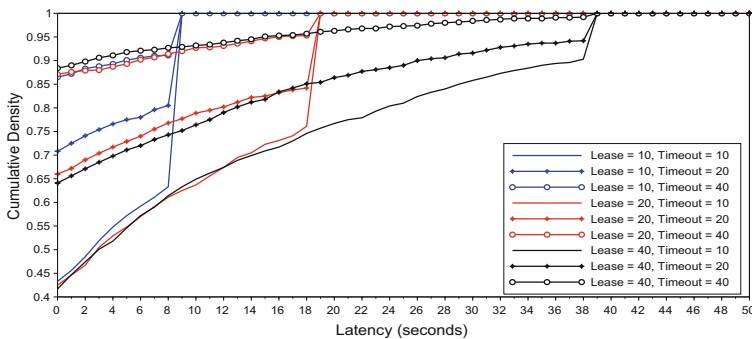


Fig. 6. Latency distributions for transactions with varying `timeout` and `lease` periods.

users access the system every 20 min on average to receive up-to-date transport information on their hand-held devices. They stay connected for a `timeout` period and then disconnect, also for resource saving purposes. Actual connection/disconnection behavior is based on the user’s profile and context at the specific time. By relying on our statistical analysis, an application designer may configure the `timeout` period of user access to 10 min. Using scaled values from Figs. 5 and 6, this guarantees that the user will receive on average 65 % of the posted notifications, within at most 8 min of latency with a probability of 0.63. If these values are insufficient and the designer re-configures the `timeout` to 20 min, this guarantees that now the user will receive on average 80 % of the posted notifications, within at most 4 min of latency with a probability of 0.77. This technique can be extended to other scenarios, where varying such parameters would provide improvements in performance metrics.

5.3 Comparison with XSB Implementation

In order to validate the simulations performed in Sect. 5.1, we implement realistic transactions using the XSB framework. Specifically, we use two middleware implementations: (i) for `lease = 0` transactions, the DPWS⁴ CS middleware provides an API to set a poster and a getter interacting with each other directly; and (ii) for (`lease > 0`) transactions, the JMS⁵ PS middleware provides an API to set a poster, a getter, and the intermediate entity through which they interact. Applying the same settings as in Sect. 5.1, posters and getters perform operations based on probability distributions (exponential δ_{post} with mean of 10 s and δ_{get} with various mean periods). At the intermediate entity we set various `lease` periods, using the JMS API. Note that in these XSB implementation settings, we have concurrent `posts` and queueing. This corresponds to an M/G/1/ ∞ queueing model; however, the queueing time of data due to processing of preceding data is negligible in our specific settings. All the transactions are performed using an Intel Xeon W3550e 3.08 GHz \times 4 (7.8 GB RAM) under a *Linux Mint* OS. For getting reliable results, the mean values of δ_{post} and δ_{get} intervals are expected to be close to the expected mean values. To do so, we create sufficient number of `post` operations

Table 3. Simulated vs measured transaction success rates.

<code>lease</code> (s)	δ_{get} (s)	Simulation	Measurement
0	<code>exponential(20)</code>	0.65	0.717
0	<code>exponential(40)</code>	0.35	0.42
10	<code>exponential(20)</code>	0.75	0.778
10	<code>exponential(40)</code>	0.48	0.554
40	<code>exponential(20)</code>	0.93	0.91
40	<code>exponential(40)</code>	0.75	0.81

⁴ <http://ws4d.e-technik.uni-rostock.de/jmeds>.

⁵ <http://activemq.apache.org>.

and `get` connections/disconnections by running each experiment for at least 2 h. In Table 3, we compare the results of simulated and measured success rates for `timeout = 20s`, $\delta_{\text{post}} = \text{Poisson}(10)\text{s}$, `lease = 0, 10, 40s` and various δ_{get} . The absolute deviation between the two is no more than 10%. This deviation may be attributed to implementation factors such as network delays and buffering at each entity (poster, getter, intermediate entity) which may affect the success rates. As this deviation is not too high, it allows developers to rely on our simulation model to tune the system.

6 Related Work

With an always increasing number of heterogeneous devices being interconnected among them and with conventional services through the Internet of Things [13], extensions to standard (client-service oriented) ESB-style bridging middleware [8] are required. The XSB connector [12], which resulted from the CHOReOS project [9], explicitly incorporates multiple interaction paradigms, including PS and TS schemes. XSB relies on protocol conversion [19], which allows reasoning about diverse interaction paradigms using the unifying XSB semantics. In our previous work [16], we extended the XSB connector with multi-dimensional QoS metrics that can incorporate timeliness, security and resource efficiency levels. However, we did not consider limited data lifetime, disconnections of peers, or asynchronous reception, as we do in this paper.

Our work upgrades middleware connectors for heterogeneous interaction paradigms with timing analysis. In [24], service composition models are studied where synchronous, asynchronous or parallel interaction may provide superior success rates under time constraints. Similar tuning of time parameters has been applied in distributed real time systems [17] for resource management, while checking end-to-end performance across multiple layers. Besides, middleware-based QoS control has been proposed by [7], where the QoS-aware adaptation and reconfiguration of systems is performed by reflective middleware. In [22], a grid quorum based publish-subscribe system is proposed to deal with delay-sensitive aspects of Internet of Things applications. In comparison, the contribution of our work is a unified timing analysis across heterogeneous middleware paradigms.

Timed automata [2] have been applied to a variety of real time system models to ensure accurate behavior under timed guards. Model checkers such as UPPAAL [6] and PRISM [18] have been proposed for timed and probabilistic properties of such systems. We make use of such tools for design time analysis of heterogeneous middleware interactions. Timed automata are used in [23] for studying fault tolerant behavior (safety, bounded liveness) in distributed asynchronous real timed systems. In [14], the transmission channels of publish-subscribe middleware are modeled using probabilistic timed automata to verify properties of supported interactions. The same authors do model-checking of publish-subscribe applications using Bogor [3] and the PRISM probabilistic model checker [14]. A closely related work is [1], where formal analysis (using colored Petri-Nets) of various types of time synchronization in distributed middleware architectures has been performed. Indeed, alternatives to simulation

based approaches, such as statistical model checking [5], may be applied in the context of our work in order to verify, for instance, probabilistic reachability properties. However, simulation techniques are needed as a starting point, in order to elicit distributions needed as inputs to statistical model checkers.

In our paper, we unify the verification of the timing behavior of multiple heterogeneous interactions using timed automata and their statistical analysis. While our prior work focused mainly on the functional interoperability or QoS upgrade of heterogeneous middleware systems, we further model here the fine-grained effect of timing thresholds on both coupled and decoupled distributed systems as well as their combinations. By leveraging the analysis of timing thresholds, designers of heterogeneous choreographies can accurately set constraints to ensure high success rates for transactions.

7 Conclusions

Timing constraints have typically been used for time-sensitive systems to ensure properties such as deadlock freeness and time-bounded liveness. In this paper, we study the XSB interoperable middleware connector from the *CHOReOS* project, by accurately modeling its timing behavior through timed automata. Verification of conditions for successful XSB transactions is done in UPPAAL in conjunction with the timing guards specified. We demonstrate that accurate setting of `lease` and `timeout` periods significantly affects the transaction success rate. By providing a fine-grained analysis of the related timing thresholds for designers of choreographies, increased probability of successful transactions can be ensured. This is crucial for accurate runtime behavior, especially in the case of heterogeneous space-time coupled/decoupled interactions with variable connectivity of peers. Furthermore, we demonstrate that the latency vs. success rate tradeoff can be suitably configured for heterogeneous choreographies. Finally, we confirm the sufficient accuracy of our results by comparing with experimental outcomes from the XSB implementation framework.

References

1. Aldred, L., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: On the notion of coupling in communication middleware. In: Meersman, R. (ed.) *CoopIS/DOA/ODBASE 2005*. LNCS, vol. 3761, pp. 1015–1033. Springer, Heidelberg (2005)
2. Alur, R., Dill, D.L.: A theory of timed automata. *Theoret. Comput. Sci.* **126**, 183–235 (1994)
3. Baresi, L., Ghezzi, C., Mottola, L.: On accurate automatic verification of publish-subscribe architectures. In: *IEEE International Conference on Software Engineering* (2007)
4. Barker, A., Walton, C.D., Robertson, D.: Choreographing web services. *IEEE Trans. Serv. Comput.* **2**, 152–166 (2009)
5. Basu, A., Bensalem, S., Bozgt, M., Delahaye, B., Legay, A.: Statistical abstraction and model-checking of large heterogeneous systems. *Int. J. Softw. Tools Techno. Transfer* **14**, 53–71 (2012)

6. Behrmann, G., David, A., Larsen, K.G.: A tutorial on uppaal 4.0. Technical report, Aalborg University, Denmark (2006)
7. Blair, G.S., Andersen, A., Blair, L., Coulson, G., Sanchez, D.: Supporting dynamic QoS management functions in a reflective middleware platform. *Proc. IEE Softw.* **147**(1), 13–21 (2000)
8. Chappell, D.A.: *Enterprise Service Bus*. O'Reilly Media, Sebastopol (2004)
9. CHOReOS. Final CHOReOS architectural style. Technical report, Large Scale Choreographies for the Future Internet (2013)
10. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.* **8**(2), 244–263 (1986)
11. Freeman, E., Hupfer, S., Arnold, K.: *JavaSpaces Principles, Patterns, and Practice*. Addison-Wesley Professional, Essex (1999)
12. Georgantas, N., Bouloukakis, G., Beauche, S., Issarny, V.: Service-oriented distributed applications in the future internet: the case for interaction paradigm interoperability. In: Lau, K.-K., Lamersdorf, W., Pimentel, E. (eds.) *ESOC 2013. LNCS*, vol. 8135, pp. 134–148. Springer, Heidelberg (2013)
13. Guinard, D., Karnouskos, S., Trifa, V., Dober, B., Spiess, P., Savio, D.: Interacting with the SOA-based internet of things: discovery, query, selection, and on-demand provisioning of web services. *IEEE Trans. Serv. Comput.* **3**, 223–235 (2010)
14. He, F., Baresi, L., Ghezzi, C., Spoletini, P.: Formal analysis of publish-subscribe systems by probabilistic timed automata. In: Derrick, J., Vain, J. (eds.) *FORTE 2007. LNCS*, vol. 4574, pp. 247–262. Springer, Heidelberg (2007)
15. Issarny, V., Bennaceur, A., Bromberg, Y.-D.: Middleware-layer connector synthesis: beyond state of the art in middleware interoperability. In: Bernardo, M., Issarny, V. (eds.) *SFM 2011. LNCS*, vol. 6659, pp. 217–255. Springer, Heidelberg (2011)
16. Kattepur, A., Georgantas, N., Issarny, V.: QoS analysis in heterogeneous choreography interactions. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) *ICSOC 2013. LNCS*, vol. 8274, pp. 23–38. Springer, Heidelberg (2013)
17. Kim, M., Stehr, M.-O., Talcott, C., Dutt, N., Venkatasubramanian, N.: Combining formal verification with observed system execution behavior to tune system parameters. In: Raskin, J.-F., Thiagarajan, P.S. (eds.) *FORMATS 2007. LNCS*, vol. 4763, pp. 257–273. Springer, Heidelberg (2007)
18. Kwiatkowska, M., Norman, G., Parker, D.: PRISM: probabilistic symbolic model checker. In: *Proceedings of Tools Session of Aachen International Muliconference on Measurement, Modelling and Evaluation of Computer-Communication Systems*, pp. 7–12 (2001)
19. Lam, S.S.: Protocol conversion. *IEEE Trans. Softw. Eng.* **14**(3), 353–362 (1988)
20. Richards, M., Monson-Haefel, R., Chappell, D.A.: *Java Message Service*, 2nd edn. O'Reilly, Sebastopol (2009)
21. Richardson, L., Ruby, S.: *RESTful Web Services*. O'Reilly, Sebastopol (2007)
22. Sun, Y., Qiao, X., Cheng, B., Chen, J.: A low-delay, lightweight publish/subscribe architecture for delay-sensitive IoT services. In: *IEEE 20th International Conference on Web Services* (2013)
23. Waszniowski, L., Krakora, J., Hanzalek, Z.: Case study on distributed and fault tolerant system modeling based on timed automata. *J. Syst. Softw.* **82**, 1678–1694 (2009)
24. Zhang, T., Ma, J., Sun, C., Li, Q., Xi, N.: Service composition in multi-domain environment under time constraint. In: *IEEE International Conference on Web Services* (2013)

Context-Driven Assessment of Provider Reputation in Composite Provision Scenarios

Lina Barakat¹(✉), Phillip Taylor², Nathan Griffiths², and Simon Miles¹

¹ King's College London, London, UK

{lina.barakat,simon.miles}@kcl.ac.uk

² University of Warwick, Coventry, UK

{Phillip.Taylor,Nathan.Griffiths}@warwick.ac.uk

Abstract. Service-oriented computing has become the de-facto way of developing distributed applications and, in such systems, an accurate assessment of reputation is essential for selecting between alternative providers. Existing methods typically assess reputation on a combination of direct experiences by the client being provided with a service and third party recommendations, but they exclude from consideration a wealth of information about the context of providers' previous actions. Such information is particularly important in composite service provision scenarios, where providers may *delegate* sub-tasks to others, and thus their success or failure needs to be interpreted in this context and reputation assessed according to responsibility. In response, to enable richer, more accurate reputation mechanisms, this paper models the delegation knowledge underlying a composite service provision, and incorporates such knowledge into the reputation assessment process, adjusting the contributions of past interactions with the composite service provider according to delegation context relevance. Experimental results demonstrate the effectiveness of the proposed approach.

Keywords: Reputation assessment · Delegation context · Composite service provider · Interaction weighting

1 Introduction

A service-oriented system can be seen as a dynamic marketplace, where individuals and organisations rely on providers to execute services with an appropriate quality in order to fulfil their own goals. Such reliance implies a degree of risk through dependence upon a third party, and so there is a need for mechanisms that ensure correctness and fairness of providers' and customers' behaviour to help assess and manage this risk. Trust and reputation are concepts commonly modelled in mechanisms for improving the success of interactions by minimising uncertainty when self-interested individuals interact [11]. Trust is an assessment of the likelihood that an individual or organisation will cooperate and fulfil its commitments [12], while reputation can be viewed as the public perception of

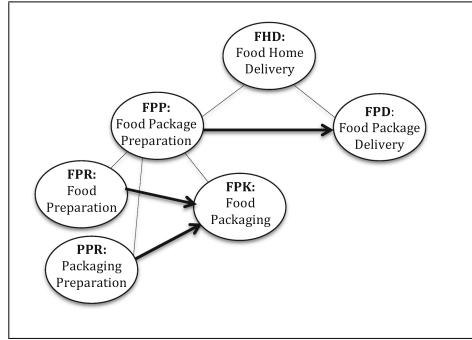


Fig. 1. Delegation hierarchy of food home delivery

the trustworthiness of a given entity [13]. In this paper, we will use the terms trust and reputation interchangeably.

Many models exist in which reputation is derived from direct experience of clients and third party recommendations, with numerical or probabilistic representations for reputation [1–7]. However, these existing methods focus solely on the clients’ experience of the ultimate outcome of a service provision, without consideration of the full history of activity behind such provision, thus omitting from assessment potentially relevant information. Such information is of particular importance in the case of composite service provision, where the provider may depend on other sub-providers to accomplish the task requested.

To illustrate, consider a provider P_{FHD} offering home delivery of nutritious food packages (e.g. meal packages) to customers. The provider does not produce the food packages locally, but deals with a number of specialised suppliers for this purpose (e.g. professional individuals, food companies, etc.). Specifically, provider P_{FHD} exposes the options available from these suppliers to potential customers via a dedicated search interface, collects a customer’s order, and delegates this order to a suitable food package supplier. Following the preparation of the food package, provider P_{FHD} contacts its food transportation partner, provider P_{FPD} , for the delivery of the food package to the customer. Figure 1 depicts the delegation hierarchy of provider P_{FHD} and its sub-providers. This hierarchy is not visible to the customer, who rates its interaction with provider P_{FHD} based on the customer’s perception of the final outcome (the food package delivered). Computing the reputation of provider P_{FHD} , without interrogating the history of delegation underlying customer ratings, might yield an inaccurate reputation assessment for the current situation. For example, a low reputation of provider P_{FHD} due to the past failures of its transportation sub-provider P_{FPD} to meet consumers’ constraints on delivery time might give an unfair view if the sub-provider has been changed to avoid such failures re-occurring.

In response, this paper extends existing reputation models by incorporating the delegation context underlying a composite service provision into the reputation assessment process¹. In particular, the delegation context of a provision

¹ See <http://jaspr.org/case-studies> for example usage scenarios.

Table 1. Symbols often used throughout the paper

Symbol	Description	Symbol	Description
<i>attr</i>	Capability’s attributes	<i>P</i>	Set of providers
<i>C</i>	Set of capabilities	<i>rating</i>	Interaction’s rating for an aspect
<i>cmp</i>	Goal (composite) capability	<i>rel</i>	Delegation context relevance
<i>end</i>	Capability’s candidate providers	<i>rep</i>	Reputation score for an aspect
<i>dctx</i>	Interaction’s delegation context	<i>time</i>	Interaction’s time
<i>I</i>	Set of interactions	<i>wt</i>	Interaction’s weight

is utilised as additional evidence for determining the relevance of the ratings available on this provision for the current situation, thus enabling more accurate and informed reputation estimates. The rest of the paper is organised as follows. Sections 2 and 3 provide the delegation model and the rating model, respectively, for a composite service provider. The incorporation of delegation context into reputation assessment is detailed in Sect. 4, while a corresponding evaluation is presented in Sect. 5. Section 6 discusses related issues, Sect. 7 presents related work, and finally Sect. 8 concludes the paper.

2 Delegation Model

To achieve a composite task *cmp*, the corresponding provider, henceforth referred to as P_{cmp} , may often rely on a number of sub-contractors to perform the various sub-tasks involved. Formally, knowledge of such a delegation model is a tuple, $(C, P, end, dg, attr)$, detailed below (see Table 1 for the often used notation).

C is the set of capabilities relevant for achieving composite task *cmp* (including *cmp* itself). Each capability $c \in C$ denotes a competence in achieving a particular sub-task (simple or composite) of composite task *cmp*. In our example, $C = \{FHD, FPP, FPD, FPR, PPR, FPK\}$. Note that, we refer to a capability and its corresponding task interchangeably. We keep the definition of a capability generic to be applicable to a wide range of domains, e.g. it may refer to an operation signature, a resource specification, or an ontology term.

P is the set of available providers in the community offering capabilities from C . Providers encapsulate such offerings within *services*, and expose them through uniform, machine-readable interfaces (or metadata) on a network of customers.

$end : C \rightarrow 2^P$, is the candidate provider function, mapping each capability $c \in C$ to the set of providers $end(c) \subset P$ offering this capability as a service. Different mechanisms are possible for discovering candidate providers: by consulting a central service repository (e.g. a UDDI² registry) storing service metadata (e.g. SAWSD³ descriptions); or by calling for service proposals over the network (e.g. using the contract net protocol [16]). We make no assumptions in our model about any specific technology or service discovery and matching mechanism.

² http://uddi.org/pubs/uddi.v3.htm#_Toc85907967.

³ <http://www.w3.org/TR/sawSDL/>.

$dg : C \rightarrow G$, is the decomposition graph function, defining the hierarchical delegation structure in the community. It maps a composite capability $c \in C$ to its decomposition graph $dg(c) = (V, E) \in G$, which specifies the comprising (finer-grained) sub-capabilities that can be outsourced to sub-contractors and their dependencies, such that $V \subset C$, $E \subset V \times V$, and G is the set of all directed graphs that can be formed from the capabilities in C . For instance, in our example, $dg(FHD) = (\{FPP, FPD\}, \{(FPP, FPD)\})$.

Finally, $attr : C \rightarrow 2^{AN}$, is the attribute function, mapping each capability $c \in C$ to the set of attributes $attr(c) \subset AN$ characterising this capability (AN is the set of all attribute names). Such features are domain dependent, and can be either *global* (common to all tasks), e.g. cost and duration, or *local* (only specific to particular tasks), e.g. the quality of packaging (which is specific to packaging-related tasks) in the food manufacturing domain. Note that a composite capability inherits all the attributes characterising its sub-capabilities.

3 Rating Model

The rating model of a composite service provider P_{cmp} (the ratee), enriched with delegation context, is a tuple, $(I, time, rater, rating, dctx)$, as detailed below.

I is the set of previous interactions with the ratee up to the current time step, for which ratings are available to the reputation assessor.

$time : I \rightarrow T \cup \{\perp\}$, is an interaction time step function, specifying for an interaction $i \in I$, the point in time $time(i) \in T$ at which the interaction took place (T is the set of all time steps). Note that $time(i) = \perp$ indicates that the time of interaction i is either not utilised for reputation assessment (the case of some reputation models), or unknown (e.g. knowledge of the interaction is acquired from a third party with no time information).

$rater : I \rightarrow U \cup \{\perp\}$, is an interaction rater function, specifying the rating party $rater(i) \in U$ for interaction $i \in I$ (U is the set of all users who interacted with the ratee). The availability and utilisation of the rater knowledge depend on the reputation model adopted: while some models allow for third party ratings, others only account for personal experience. Moreover, in some models, the rater identities are kept anonymous with the ratings being collected and accessible via a dedicated store, in which case $\forall i \in I, rater(i) = \perp$.

$rating : I \times AN \cup \{\text{overall}\} \rightarrow RV \cup \{\perp\}$, is an interaction rating function, mapping an interaction $i \in I$, to the rating, $rating(i, a) \in RV$, provided for this interaction on aspect $a \in attr(cmp) \cup \{\text{overall}\}$. Here, *overall* denotes an overall perspective on the interaction, and RV is the set of all possible rating values. Note that $rating(i, a) = \perp$ indicates that rating on aspect a is not available for interaction i . The domain of rating values RV depends on the reputation model adopted, e.g. it can be binary, indicating either success or failure, or numeric, indicating the level of satisfaction according to a particular scale.

Finally, $dctx : I \times C \rightarrow P \cup \{\perp\}$, is an interaction's *delegation context* function (which we introduce into the rating model), representing knowledge regarding the participants in the realisation of composite capability cmp during interaction $i \in I$. In particular, it maps a sub-capability node $c \in C$ in a decomposition

graph, to the provider $dctx(i, c) \in cnd(c)$, to which this capability was delegated by the ratee (or its sub-contractors) during interaction i . Details of how such delegation knowledge can be acquired by the reputation assessor are discussed in Sect. 6. In general, different levels of visibility might be available to the assessor regarding this knowledge. For example, the assessor might be aware of the delegation hierarchy fully, or only partially (e.g. only the direct sub-contractors of the ratee, performing the sub-capabilities of $dg(cmp)$, are known), or as in the traditional rating model, might not have access to any delegation knowledge except for $dctx(i, cmp) = ratee$, i.e. $\forall c \in C \setminus \{cmp\}, dctx(i, c) = \perp$.

4 Context Exploitation for Reputation Assessment

A basic abstraction of a number of existing reputation assessment models is a tuple (wt, rep) , as detailed below.

$wt : I \rightarrow WV$, is an interaction weighting function, governing the contribution of each available previous interaction $i \in I$ for the reputation assessment at hand. Here, WV is the set of possible weight values, and can be a binary domain, corresponding to an interaction selection decision, or a continuous domain, corresponding to an interaction ranking decision. Commonly, the factors playing a role in the evaluation of an interaction's weight $wt(i)$, are its recency $time(i)$ (recent observations are usually favoured over older ones), and the observing party $rater(i)$ (direct experience is usually favoured over third party opinions).

$rep : AN \cup \{\text{overall}\} \rightarrow PV$, is the reputation assessment function, providing a numeric value $rep(a) \in PV$ that reflects the level of trust the assessor places in the ratee (at the current time point) with respect to aspect $a \in attr(cmp) \cup \{\text{overall}\}$ of capability cmp . Depending on the reputation model and the domain of rating values, $rep(a)$ may refer to the expected probability of success, the expected rating, etc. It is usually estimated by applying some statistical summary function, **aggr**, over the ratings of interactions $i \in I$, while accounting for their weights $wt(i)$, i.e., $rep(a) = \mathbf{aggr}(\{\langle wt(i), rating(i, a) \rangle\}_{i \in I})$. For example, **aggr** may correspond to a weighted mean over numeric ratings, or a probability estimation measure over categorical (e.g. binary) ratings, etc.

The extension we propose to existing reputation models is to account for an interactions's delegation context, $dctx(i, c \in C)$, in the assessment of its weight $wt(i)$. The idea is to assign higher weights to those interactions sharing similar delegation context with the potential interaction under consideration. The intuition behind this is straightforward: the best indication of how a provider may behave in future is how the provider behaved under *similar conditions* in the past, with differing conditions potentially leading to different behaviour. To achieve this, we are interested in quantifying the relevance between the delegation context of each existing interaction $i \in I$, and that of the potential future interaction, which we refer to as fi . Details are presented next.

4.1 Delegation Context Relevance Assessment

We aim for four criteria to be accounted for by the measure of relevance between the delegation contexts of interactions $i \in I$, and that of future interaction fi .

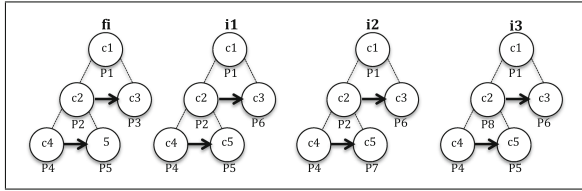


Fig. 2. Multi-level relevance: i_1 is more relevant than i_2 and i_3

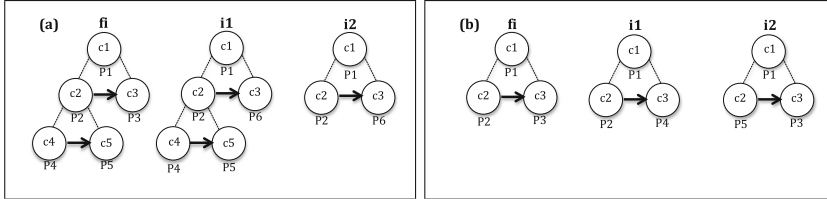


Fig. 3. Hidden Deleg. Context (a), Attr. centric Cap. (b): i_1 is more relevant than i_2

1. *Multi-level relevance*, taking into consideration similarities of delegation context across all levels of the delegation hierarchy. For example, in Fig. 2, i_1 should be considered more relevant to fi than both i_2 and i_3 . This is because, when compared to i_1 , i_2 further differs from fi in the leaf provider of capability c_5 , while i_3 further differs in the intermediary provider of capability c_2 . In fact, whilst it is important to account for differences at the level of leaf sub-contractors since these are the ones actually performing the requested capabilities, intermediary sub-contractors may also play an important role in coordination, passing on truthful information, etc.
2. *Hidden delegation context*, discounting the relevance of an interaction in the case of uncertainty (i.e. missing information) regarding its delegation context. For example, in Fig. 3(a), i_1 should be considered more relevant to fi than i_2 . This is because, unlike i_1 , i_2 's instantiation of capabilities c_4 and c_5 could be different from those of fi .
3. *Attribute-centric capabilities*, assigning higher importance to those capabilities in the delegation hierarchy that have an impact on the attribute under assessment. For instance, consider Fig. 3(b) and assume that capabilities c_1 , c_2 , and c_3 correspond to *FHD*, *FPP*, and *FPD* of our motivating example, and that the attribute under assessment is *portion size* of the delivered food. Both i_1 and i_2 differ from fi in one provider, but i_1 should be considered more relevant to fi than i_2 . This is because, since food package delivery (*FPD*) has no effect on *portion size*, which is mainly determined by food package preparation (*FPP*), any difference in the performances of providers P_3 and P_4 would not cause any deviation for this attribute, as opposed to a difference between P_2 and P_5 , which could potentially affect *portion size*.

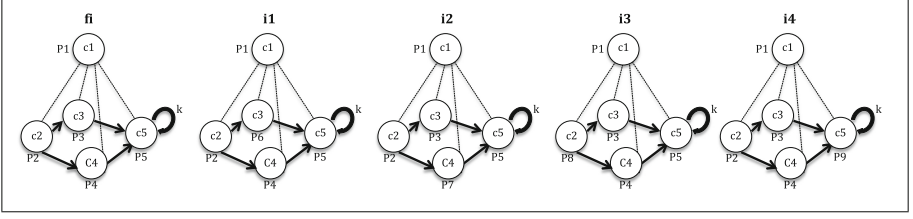


Fig. 4. Composition structure: i_1 is more relevant than i_2 ; i_3 is more relevant than i_4

4. *Composition structure*, taking into consideration the various connectivity constructs (e.g. sequence, parallel, loop) among sub-capabilities in a decomposition graph. For instance, consider Fig. 4, where capabilities c_3 and c_4 are performed in parallel, and c_5 is repeated k times. Assume assessment of *execution time*, with the domain knowledge indicating that c_4 usually takes much longer to be achieved than c_3 . Whilst both i_1 and i_2 differ from fi in one provider in the parallel construct, i_1 should be considered more relevant than i_2 to fi . This is because, given that c_3 's execution time is normally dominated by that of c_4 regardless of the provider, any difference in the performances of providers P_3 and P_6 is unlikely to affect the overall execution time observed for the interaction, as opposed to a similar difference in the performances of P_4 and P_7 . Similarly, i_3 should be considered more relevant to fi than i_4 since a difference in the performances of P_9 and P_5 would be magnified k times, unlike a similar difference between P_8 and P_2 .

A simple measure of relevance, $rel \in [0, 1]$, satisfactory of the above, between the delegation context of interaction i , $dctx(i, c \in C)$ (referred to as $dctx_i$ for simplicity), and the delegation context of future interaction fi , $dctx(fi, c \in C)$ (referred to as $dctx_{fi}$), with respect to attribute $a \in attr(cmp)$, can be given as:

$$rel(dctx_i, dctx_{fi}, a) = \sum_{\substack{c \in C, \\ dctx(fi, c) \neq \perp}} role(c, a) \times prel(dctx(i, c), dctx(fi, c)) \quad (1)$$

Function $role(c, a) \in [0, 1]$ defines the attribute-dependent distribution of roles among the capabilities under assessment (those *instantiated* with a provider in fi). That is, it specifies the relative importance of a capability for the relevance assessment, s.t. $\sum_{c \in C, dctx(fi, c) \neq \perp} role(c, a) = 1$. In particular, the role of capability

c regarding attribute a is determined according to its hierarchical importance, W_h , structural importance, W_s , and attribute-related importance, W_a , as:

$$role(c, a) = \frac{W_h(c) \times W_s(c, a) \times W_a(c, a)}{\sum_{\substack{c_j \in C, \\ dctx(fi, c_j) \neq \perp}} W_h(c_j) \times W_s(c_j, a) \times W_a(c_j, a)} \quad (2)$$

The hierarchical importance of capability c , $W_h(c) \in [0, 1]$, is governed by c 's level in the *instantiated* delegation hierarchy of fi . Alternative options include: (a) assigning equal importance to all levels, i.e. $W_h(c) = 1$ ($\forall c$); (b) favouring capabilities at lower levels, e.g. $W_h(c) = \frac{lvl(c)}{maxlvl(c)}$, where $lvl(c)$ is the level of c in the delegation hierarchy (with $lvl(cmp) = 1$), and $maxlvl(c)$ is the number of levels in fi 's longest *instantiated* hierarchy path containing capability c ; and (c) accounting only for the capabilities at the lowest level, i.e. $W_h(c) = 1$ if c is a leaf capability in fi 's *instantiated* delegation hierarchy, and $W_h(c) = 0$ otherwise.

The structural importance of capability c regarding attribute a , $W_s(c, a) \in \mathbb{Z}^+$, is governed by the position of c and its ancestors, $ancestor(c)$, in the corresponding decomposition graphs, and can be defined as follows:

$$W_s(c, a) = \prod_{c_j \in \{c\} \cup ancestor(c)} localW_s(c_j, a) \quad (3)$$

with $localW_s(c_j, a) = occur(c_j, unfold(critical(cg(c_j), a)))$. Here: $occur(c, g)$ is the number of occurrences of capability c in graph g ; $cg(c)$ is the decomposition graph containing capability c in the delegation hierarchy; $critical(g, a)$ returns the sub-graph of g that is considered critical for attribute a (i.e. the subgraph determining the performance regarding attribute a); and $unfold(g)$ returns the unfolded version of graph g using loop unfolding [14].

The attribute-related importance of c regarding attribute a , $W_a(c, a) \in [0, 1]$, can be defined as follows: $W_a(c, a) = 1$ if $a \in attr(c)$; and $W_a(c, a) = 0$, otherwise.

Finally, function $prel(dctx(i, c), dctx(fi, c)) \in [0, 1]$ measures the relevance between providers $dctx(i, c)$ and $dctx(fi, c)$, responsible for performing c in i and fi , respectively. It can be defined as follows:

$$prel(dctx(i, c), dctx(fi, c)) = \begin{cases} 1 & \text{if } dctx(i, c) = dctx(fi, c) \\ 0.5 & \text{if } (dctx(i, c) \neq dctx(fi, c)) \wedge (dctx(i, c) = \perp) \\ 0 & \text{if } (dctx(i, c) \neq dctx(fi, c)) \wedge (dctx(i, c) \neq \perp) \end{cases} \quad (4)$$

The correspondence between the criteria outlined earlier and the suggested relevance equation rel can be summarised as follows. The *multi-level relevance* factor is captured via the aggregation function (the sum function in Eq. 1) over the scores of relevant capabilities, as well as via the hierarchical importance component W_h of function *role*. The *hidden delegation context* factor is captured via the provider relevance function $prel$, assigning lower relevance value (i.e. 0.5) in the case of missing (unavailable) instantiation of a capability. Finally, both the *attribute-centric capabilities* and the *composition structure* factors are captured in function *role*, via the attribute-related importance component W_a and the structural importance component W_s , respectively.

4.2 Reputation Model Extension

In this Section, we show how an existing reputation model, FIRE [4], can be extended to account for the delegation context relevance proposed in Eq. 1. FIRE

combines four different types of reputation and trust: interaction trust from direct experience, witness reputation from third party reports, role-based trust, and certified reputation based on third-party references. We do not consider the role-based and certified reputation components in this paper.

Reputation is assessed in FIRE from tuples of form $(\alpha, \beta, a, i, rating(i, a))$, where α and β are agents that participated in interaction i such that α gave β a rating value of $rating(i, a) \in [-1, +1]$ for the term a . A rating of $+1$ is absolutely positive, -1 is absolutely negative, and 0 is neutral. To determine direct reputation of agent β for term a , an assessing agent α extracts the set of ratings from its database of the form $(\alpha, \beta, a, -, -)$ where “ $-$ ” matches any value. Moreover, agents maintain a list of acquaintances, and use these to identify witnesses to evaluate witness reputation. Specifically, an evaluator α will ask its acquaintances for ratings of β for term a (i.e. ratings of the form $(-, \beta, a, -, -)$). Finally, the overall trust is calculated as a weighted mean of each of the component sources.

Since we focus on investigating the effect of delegation context on reputation, we do not consider the effect of the rating party or the interaction topology in this paper. That is, for simplicity, we assume that all previous interactions with agent β are accessible to any reputation assessor (e.g. via a dedicated rating repository), assigning equal importance to all raters. In FIRE, this corresponds to a fully connected agent network, with equal weights being assigned to the individual and witness experience. Thus, to determine the reputation of a provider P_{cmp} on behalf of a client, the assessor queries the rating store for ratings of the form $(-, P_{cmp}, a, i, rating(i, a))$. These ratings are scaled using a *recency factor*, λ , in the interaction weight function, instantiated in FIRE per interaction i as:

$$wt(i) = recency(i) = e^{\frac{|time(i) - time(fi)|}{\lambda}} \quad (5)$$

The reputation value the assessor has in P_{cmp} for term a is then calculated as the weighted mean of the available ratings: $rep(a) = \frac{\sum_{i \in I} wt(i) \times rating(i, a)}{\sum_{i \in I} wt(i)}$. Note that, to combine the reputation of different attributes a into a single composite assessment for agent P_{cmp} , we use a weighted sum across all attributes: $rep(overall) = \sum_a rep(a) \times attrwt(a)$, where $attrwt(a)$ corresponds to the weight of attribute a for the client, such that $\sum_a attrwt(a) = 1$.

Now, in order to account for the delegation context in FIRE, we adjust the weighting that is given to an interaction i so that it becomes attribute dependent and incorporates delegation context relevance, as follows:

$$wt(i, a) = recency(i) \times rel(dctx_i, dctx_{fi}, a). \quad (6)$$

5 Experiments and Results

This section presents an empirical evaluation of the proposed delegation-context-aware reputation framework, focusing on its performance in terms of producing more accurate reputation assessments⁴. The simulation involves one composite

⁴ Source code and data for the results presented in this paper are freely available from <http://jaspr.org/source-code>.

provider agent interacting with a number of customers. In particular, we adopt our example scenario, showing the results from the perspective of the food home delivery provider P_{FHD} , with the delegation hierarchy of Fig. 1. We assume that each (sub-)capability can be delegated to *pnum* alternative providers.

The simulation proceeds on the basis of rounds, each corresponding to an interaction between a customer and composite provider P_{FHD} . In each round, provider P_{FHD} instantiates its delegation hierarchy with a particular combination of sub-contractors, and accordingly delivers particular values for the aspects of interest. The customer then rates provider P_{FHD} on each aspect according to their satisfaction. The customer ratings and the provider's delegation context of the current round are utilised by the reputation assessor to adjust the reputation of provider P_{FHD} for the next round. Other experimental settings are outlined in Sects. 5.1 and 5.2, followed by experimental results in Sect. 5.3.

5.1 Customer Rating Generation

The evaluation considers three attributes: execution time (ex), quality of packaging (qp), and portion size (pz) of the delivered food. We assume that the performance of provider P_{FHD} with respect to each of these attributes is determined by the corresponding performances of the leaf sub-contractors who actually perform the capabilities (therefore we utilise option (c) for weights W_h).

Assuming knowledge that the food preparation (FPR) normally takes much longer than the packaging preparation (PPR), i.e. PPR does not belong to the critical path for evaluating ex , and that the packaging could be damaged during food packaging (FPK) or during food package delivery (FPD), the values delivered by provider P_{FHD} in an interaction for each considered attribute are:

$$\begin{aligned} val_{prv}(P_{FHD}, ex) &= val_{prv}(P_{FPR}, ex) + val_{prv}(P_{FPK}, ex) + val_{prv}(P_{FPD}, ex) \\ val_{prv}(P_{FHD}, qp) &= \min(val_{prv}(P_{PPR}, qp), val_{prv}(P_{FPK}, qp), val_{prv}(P_{FPD}, qp)) \\ val_{prv}(P_{FHD}, pz) &= val_{prv}(P_{FPR}, pz) \end{aligned}$$

Here, P_c denotes the provider selected for executing capability c in the interaction, and $val_{prv}(P_c, a)$ is the value produced by provider P_c for attribute a during the interaction. The generation of values val_{prv} for atomic providers is governed by their *attribute policies*, which are represented as normal distributions (with mean μ and variance σ^2) over the corresponding attribute domains, and assigned per candidate atomic provider at the beginning of experiments.

Based on this, the utility *perceived* by the customer in an interaction regarding aspect $a \in \{ex, qp, pz\}$ is: $utility_{prv}(a) = val_{prv}(P_{FHD}, a)$, while the utility considered *acceptable* by the customer is: $utility_{acc}(a) = val_{acc}(a)$, where $val_{acc}(a)$ corresponds to the value considered acceptable for attribute a (fixed among all customers in our experiments). Given this, the rating assigned by the customer for aspect a in interaction i , $rating(i, a) \in [-1, +1]$, compatible with the reputation model discussed in Sect. 4.2, equals to:

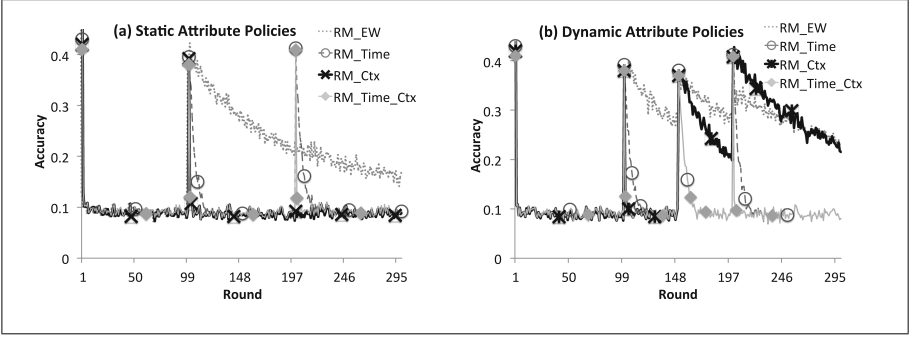


Fig. 5. Effect of Delegation Context Exploitation

$$rating(i, a) = \begin{cases} \frac{utility_{prv}(a) - utility_{acc}(a)}{\max(a) - utility_{acc}(a)} & \text{if } utility_{prv}(a) \geq utility_{acc}(a) \\ \frac{utility_{prv}(a) - utility_{acc}(a)}{utility_{acc}(a) - \min(a)} & \text{otherwise} \end{cases}$$

where $\min(a)$ and $\max(a)$ are the minimum and maximum possible values for a .

Finally, the overall rating of provider P_{FHD} in interaction i is given as: $rating(i, overall) = \sum_{a \in \{ex, qp, pz\}} attrwt(a) \times rating(i, a)$, where $attrwt(a)$ is the weight of attribute a for the customer (fixed to $\frac{1}{3}$ for the three attributes).

5.2 Evaluation Strategies and Measure

We refer to the following reputation strategies: RM_EW , the reputation model assigning equal weights to all interactions, i.e. $\forall i, wt(i, a) = 1$; RM_Time , the reputation model weighting interactions according to recency, i.e. according to Eq. 5; RM_Ctx , the reputation model weighting interactions according to delegation context relevance, i.e. according to Eq. 6 with $\forall i, recency(i) = 1$; and RM_Time_Ctx , the reputation model weighting interactions according to both recency and delegation context relevance, i.e. according to Eq. 6. As a performance measure, we quantify the difference between the provider's reputation exposed to the customer prior to an interaction i , $rep^i(a)$ (which, given the reputation model adopted, can be viewed as the predicted rating for the interaction), and the customer's actual rating following the interaction, $rating(i, a)$. That is, we measure $|rep^i(a) - rating(i, a)|$ at each round (with $a = overall$).

5.3 Results

In this section, we compare the outlined strategies under various environment settings. All the results reported are averaged over 100 simulation runs.

The Effect of Delegation Context Exploitation. Figure 5(a) reports the results in settings with *dynamic* delegation context, and assuming *static* attribute policies of sub-providers. In particular, composite provider P_{FHD} changes its delegation context (i.e. switches its sub-providers to different ones),

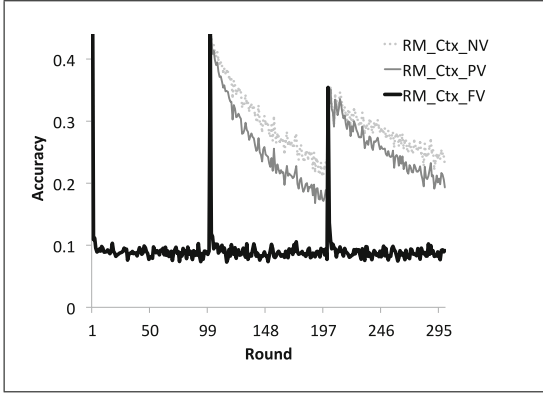


Fig. 6. Effect of Visibility Levels of Delegation Context

then returns back again to the old delegation context after another 100 rounds. As can be seen, *RM_EW* suffers from poor accuracy after the first change in the delegation context, since the reputation score mostly reflects old, no longer relevant ratings. This degradation in performance is of less severity after the second (recurring) change, with the ratings observed during the first 100 rounds becoming relevant again. Strategy *RM_Time* achieves better adaptive behaviour by favouring more recent ratings and gradually forgetting outdated ones, but is outperformed by *RM_Ctx* (the best performing strategy in this case). This is because, by utilising knowledge of delegation context, *RM_Ctx* incorporates only the most relevant ratings into reputation assessment (eliminating irrelevant ratings, collected under different delegation context). Thus, it achieves the fastest recovery of accuracy after the first change, and avoids an accuracy drop after the second change (by favouring the ratings collected in the first 100 rounds, which become relevant again). No further performance improvement is achieved by combining delegation context awareness with recency in this case.

Figure 5(b) reports the results with *dynamic* delegation context settings as above, but also with *dynamic* attribute policies of sub-providers. In particular, the attribute policies of each candidate sub-provider are set to change after 150 rounds, with a policy change being simulated by a repositioning of the corresponding mean μ . Here, although *RM_Ctx* still achieves dominating results after the change of the delegation context at round 100, it fails to do so once the providers' policies change at round 150, and further deteriorates in performance when the old delegation context reoccurs at round 200. This is because, *RM_Ctx* considers all the ratings collected under similar delegation context to be of equal importance, despite the fact that those collected prior to round 150 may no longer be relevant. The best performing strategy in this case is *RM_Time.Ctx*, which can eliminate the effect of such irrelevant ratings with time, while keeping the advantages of delegation context utilisation.

The Effect of Delegation Knowledge Granularity. Figure 6 compares the performance of *RM_Ctx* under various delegation context visibility levels, in settings with *dynamic* delegation context as above and *static* attribute policies of

sub-providers. In particular, we compare: *RM_Ctx_FV*, assuming full visibility of the delegation context; *RM_Ctx_PV*, assuming partial visibility of the delegation context, where only the direct sub-contractors, P_{FPP} and P_{FPD} , of composite provider P_{FHD} are known; and *RM_Ctx_NV*, assuming no visibility of the delegation context, i.e. the only provider visible is provider P_{FHD} . Changes in delegation context are assumed to only affect leaf capabilities FPR , PPR , FPK , and FPD , while the intermediary provider of capability FPP always remains the same. Clearly, *RM_Ctx_NV* does not detect delegation context changes, assigning equal weights to all ratings for the duration of the simulation, thus exhibiting bad performance after change points. *RM_Ctx_PV*, on the other hand, is only able to observe the change in the sub-provider of leaf capability FPD . As a result, it discounts the importance of the ratings before the change, but does not eliminate their effect entirely (these ratings are still considered partially relevant), which decreases its accuracy compared to *RM_Ctx_FV*.

6 Discussion

Why would providers expose (true) delegation context? Providers are the obvious source of delegation context as it is a record of how they provided a service, but it may be against their interests to release such records. There are a few initial answers to this question, though full exploration of the issue is beyond the scope of this paper. First, such information should be expected to be present in the client-accessible service advert at the time of service provision. Second, there are two agents in an interaction that could provide (or verify) information regarding a particular delegation, the delegator and the delegatee (a commonly used mechanism for non-repudiation). Finally, the contracts which clients agree with providers can require some recording of details as part of service provision, possibly with involvement of a notary to help ensure validity.

How would providers expose delegation context? The PROV standard [10] (published by W3C as a standard for interoperable provenance) could provide a suitable solution for this purpose [15]. A PROV document describes in a queryable form the causes and effects within a particular past process of a system (such as agents interacting, the execution of a program, or enactment of a physical world process), as a directed graph with annotations. The contents of a provenance graph can be collated from data recorded by a set of independent agents, and clients have a standard means to query the data, e.g. by SPARQL⁵.

7 Related Work

In a service-oriented system individuals and organisations rely on providers to successfully execute services with an appropriate quality to fulfil their own goals, and such reliance implies a degree of risk. Trust and reputation provide an effective way of assessing and managing this risk, and are studied by researchers from many domains. In multi-agent systems, most established computational

⁵ <http://www.w3.org/TR/sparql11-overview/>.

reputation models, such as TRAVOS [1], HABIT [2], ReGreT [3] and FIRE [4], typically use a combination of direct and indirect experience. In TRAVOS [1], the trust score is the expected probability (using the beta distribution) that the trustee will fulfil its obligations towards the truster in an interaction, estimated based on the outcomes of the previous direct interactions with the trustee. When there is a lack of personal experience, the truster seeks the opinions of other sources, accounting for their reliability. Similarly, HABIT [2] uses a probabilistic approach, utilising Bayesian network to support reasoning about reputation. ReGreT [3] takes into account three dimensions of reputation: the individual dimension (based on direct interactions), the social dimension (from other sources utilising the group relation), and the ontological dimension (defining the different reputational aspects). FIRE [4] (adopted in this paper) is based on ReGreT, adding role-based trust and certified reputation based on third-party references.

Trust and reputation models have also been investigated in service-oriented systems. For example, Maximilien et al. [5] estimate a service's reputation for a quality by aggregating its previously observed quality values (shared and accessible to all assessors). Similarly, Xu et al. [6] extend the UDDI registry with a reputation manager, aggregating the past ratings of a service into a reputation score. Malik et al. [7] propose a decentralised approach for service reputation assessment, where customers seek ratings from their peers, with the credibility of ratings being estimated based on deviation from the majority opinion.

These approaches view a service as a simple service, ignoring potential composition information behind its provision, and relying mainly on recency to discount the effect of irrelevant past interactions. We argue that recency alone is not sufficient as the behaviour of a composite service is affected by its underlying composition circumstances, which may change, or older ones reoccur (making older interactions better predictors of the current service behaviour). To provide more accurate indications of interaction relevance, our work complements such recency-based interaction weighting with composition-context-based weighting.

Finally, a number of researchers focus on designing suitable mechanisms for distributing the score obtained by a composite service to its component services [8,9]. Proposed factors for governing such distribution include a component service's structural importance, replaceability, and run-time performance. However, these approaches do not account for the implications of the component services on the reputation evaluation mechanism of the composite service itself (the focus of this paper), but can be considered complementary to our work.

8 Conclusion

This paper presented how delegation information underlying a composite service provision can be utilised to provide more accurate reputation assessment of the composite service provider. Specifically, such information is used to scale the ratings available for the provider, assigning higher weights to those collected under circumstances comparable to the current settings. The proposed composition-context-based weighting is independent of any particular reputation

model, but for evaluation purposes, was incorporated into an existing reputation model, FIRE, which scales ratings according to recency. The results show that it results in improving performance. Future work involves accounting for alternative decomposition graphs per capability, and for personalised user requirements.

Acknowledgments. This work was part funded by the UK Engineering and Physical Sciences Research Council as part of the Justified Assessments of Service Provider Reputation project, ref. EP/M012654/1 and EP/M012662/1.

References

1. Teacy, W.T.L., Patel, J., Jennings, N.R., Luck, M.: Coping with inaccurate reputation sources: experimental analysis of a probabilistic trust model. In: 4th International Conference on Autonomous Agents and Multiagent Systems, pp. 997–1004 (2005)
2. Teacy, W.T.L., Luck, M., Rogers, A., Jennings, N.R.: An efficient and versatile approach to trust and reputation using hierarchical bayesian modelling. *Artif. Intell.* **193**, 149–185 (2012)
3. Sabater-Mir, J., Sierra, C.: Regret: a reputation model in gregarious societies. In: 4th Workshop on Deception, Fraud and Trust in Agent Societies, pp. 61–69 (2001)
4. Huynh, T.D., Jennings, N.R., Shadbolt, N.R.: An integrated trust and reputation model for open multi-agent systems. *J. Auton. Agent. Multi-Agent Syst.* **13**, 119–154 (2006)
5. Maximilien, E.M., Singh, M.P.: Agent-based trust model involving multiple qualities. In: 4th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 519–526 (2005)
6. Xu, Z., Martin, P., Powley, W., Zulkernine, F.: Reputation-enhanced QoS-based web services discovery. In: IEEE International Conference on Web Services, pp. 249–256 (2007)
7. Malik, Z., Bouguettaya, A.: RATEWeb: reputation assessment for trust establishment among web services. *VLDB J.* **18**, 885–911 (2009)
8. Nepal, S., Malik, Z., Bouguettaya, A.: Reputation propagation in composite services. In: IEEE International Conference on Web Services, pp. 295–302 (2009)
9. Wen, S., Li, Q., Yue, L., Liu, A., Tang, C., Zhong, F.: CRP: context-based reputation propagation in services composition. *SOCA* **6**, 231–248 (2012)
10. W3C. PROV model primer (2013). <http://www.w3.org/TR/prov-primer/>
11. Pinyol, I., Sabater-Mir, J.: Computational trust and reputation models for open multi-agent systems: a review. *Artif. Intell. Rev.* **40**, 1–25 (2013)
12. Gambetta, D.: Can we trust trust? In: Gambetta, D. (ed.) *Trust: Making and Breaking Cooperative Relations*, pp. 213–237. Basil Blackwell, Oxford (1988)
13. Jsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. *Decis. Support Syst.* **43**, 618–644 (2007)
14. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for web services composition. *IEEE Trans. Softw. Eng.* **30**, 311–327 (2004)
15. Miles, S., Griffiths, N.: Incorporating mitigating circumstances into reputation assessment. In: 2nd International Workshop on Multiagent Foundations of Social Computing (2015)
16. Smith, R.G., Davis, R.: Frameworks for cooperation in distributed problem solving. *IEEE Trans. Syst. Man Cybern.* **11**, 61–70 (1981)

Data Services and Cloud Platform Management

Runtime Model-Based Privacy Checks of Big Data Cloud Services

Eric Schmieders^(✉), Andreas Metzger, and Klaus Pohl

paluno (The Ruhr Institute for Software Technology),
University of Duisburg-Essen, Essen, Germany
{eric.schmieders, andreas.metzger, klaus.pohl}@paluno.uni-due.de

Abstract. Cloud services have to comply with privacy policies when storing or processing data. As cloud services become increasingly data-intensive, e.g., in the case of big data analytics, data privacy concerns become more critical and challenging to address. In particular, data may only be processed at certain geo-locations. However, the actual geo-locations of the many storage and compute nodes involved in big data processing is dynamically selected during runtime. In addition, the execution of concrete data processing tasks may change data classifications from, e.g., personal to anonymized data. Thus, privacy policy checks for big data cloud services have to consider information about the actual nodes and data processing tasks at runtime. The proposed approach R-PRIS monitors cloud services to derive and maintain typed runtime models providing the aforementioned information. R-PRIS checks the typed runtime models against privacy policies by employing a data-classification-aware search. The evaluation of R-PRIS, performed on Amazon Web Services (including Hadoop), indicates that the approach may efficiently and timely detect privacy violations in big data cloud services.

Keywords: Privacy · Big data · Cloud services · Runtime checking

1 Introduction

Cloud services have to comply with privacy policies when storing, transferring, and processing data. For instance, the EU Data Protection Directive¹ (DPD) as well as the US Health Insurance Portability and Accountability Act² (HIPAA) only permit processing personal data within countries that implement sufficient data protection mechanisms. Moreover, privacy policies, such as the ones proposed by NIST 800-122 or FIPS 199³, distinguish between different data classifications. Data classifications indicate the data's identifiability or sensitivity, which requires to treat the classified data accordingly.

¹ <http://eur-lex.europa.eu/>.

² <http://www.hhs.gov/ocr/privacy/>.

³ <http://csrc.nist.gov/>.

As cloud services become increasingly data-intensive – being used for large-scale and real-time big data analytics tasks for instance [3] – the implementation of such privacy policies becomes ever more challenging. Data and processing tasks are distributed among a vast number of storage and compute nodes to cope with the high volume of data and to ensure the high velocity of data processing (e.g., when using the MapReduce programming model). In addition, data classifications may dynamically change based on the data processing tasks executed by the various compute nodes (e.g., a task may aggregate personal customer data into anonymized sales statistics). On top of that, storage and compute nodes may be dynamically deployed, replicated, and migrated to achieve performance, availability, and cost goals of cloud providers. Given all this complexity and dynamism, each of the involved nodes still has to comply with privacy policies during the entire cloud service life-cycle.

Existing approaches for checking privacy policies have not addressed cloud elasticity or data classification changes (e.g., [6, 7, 10, 13]). In our previous work [16, 17], we introduced R-PRIS, a privacy compliance checking approach for dynamic cloud services. In this paper we extend R-PRIS to cope with the aforementioned challenges imposed by *data-intensive* services. In particular, the extended R-PRIS approach is able to consider data classification changes. This is important, as disregarding data classification changes may lead to a high rate of false positive violations. Such a high positive rate would limit applicability for big data cloud services. For instance, it would prohibit migrating many of the storage and compute nodes that do not process privacy-relevant data.

The extended R-PRIS approach utilizes a data-classification-aware search strategy based on typed runtime models. To this end, we propose typed runtime models for reflecting data placement and data classification changes. In order to facilitate the specification of fine-grained privacy policies, R-PRIS allows for defining data classifications, impact levels, and their relations. Using this information, R-PRIS monitors the cloud services, automatically updates the typed runtime model, and checks the model against privacy policies.

We evaluated the applicability and performance of R-PRIS for a data-intensive cloud service hosted on Amazon Web Services (AWS), leveraging Hadoop clusters for data processing. Our results indicate that R-PRIS is able to efficiently and timely detect privacy violations in big data cloud services.

The remainder of the paper is structured as follows: Sect. 2 identifies relevant classes of privacy policy violations. Using these classes of violations as basis, Sect. 3 describes the R-PRIS approach. Section 4 investigates the realizability of the approach by means of a proof of concept implementation. Section 5 describes the setup and results of experiments to evaluate the performance of R-PRIS. Related work is discussed in Sect. 6.

2 Privacy Policy Violations in Big Data Cloud Services

Cloud providers allow for specifying the geo-locations at which virtual machines shall be executed. However, misconfigured cloud infrastructures, software failures, and incorrect geo-location specifications might lead to virtual machine placements that result in privacy policy violations.

As a simple example for such situations, take two interacting service components $v1$ and $v2$ that process personal data. Both components are deployed on separate virtual machines that are initially hosted on cloud data centers within the EU. During runtime, the virtual machine hosting $v1$ is migrated to a data center outside of the EU. After migration, $v1$ and $v2$ continue to exchange personal data. However, as this now implies transfer of data beyond EU borders, this may violate data geo-location policies.

To identify these dynamic changes that need to be reflected in the R-PRIS runtime models, we systematically determine changes of cloud services properties. We focus on changes that stem from the cloud service software architecture and the underlying cloud infrastructure. We analyze whether the identified changes can lead to privacy policy violations. To be specific, we follow three steps in our analysis: (i) we identify architectural, deployment, and functional *properties* of big data cloud services that – if changed – may lead to policy violations (column 1 in Table 1); (ii), we determine concrete types of *changes* of these properties; e.g., due to cloud elasticity mechanisms (column 2); and (iii) we identify the concrete *violations* of privacy policies resulting from these changes (column 3). Table 1 summarizes the results of the analysis.

Table 1. Policy violations in big data cloud services

#	Property	Change	Violation
1	Compute node's geo-location	New node; replication; migration	A compute node is instantiated at an excluded geo-location (e.g., a virtual machine hosting a marketing application is instantiated in the US)
2	Storage nodes geo-location	Data upload; replication; migration	A storage node is assigned to store personal data excluded at the node's geo-location (e.g., a Hadoop node has to process a new chunk of personal data)
3	Node interactions	New node; migration; replication	A new interaction between nodes is established that involve transfer of personal data to excluded geo-location (e.g., an existing Hadoop cluster is integrated into a cloud service)
4	Data classifications	Changed processing task	The new task may produce data with different classification (e.g., a Hadoop task determines the average number of purchased items for a specific customer)

3 R-PRIS: Privacy Checks for Big Data Cloud Services

The main idea of R-PRIS (*Runtime model-based PRIVacy checkS*) is to utilize runtime models for performing privacy policy checks. In general, runtime models

are dynamically updated abstractions of the reflected systems [18]. The architectural runtime models of R-PRIS contain information about compute and storage node deployments, data flows occurring among these nodes, and data classification changes. The main steps of R-PRIS are depicted in Fig. 1: cloud *monitoring* information is used for *updating* architectural runtime models, which in turn are employed for policy violation *checks*.

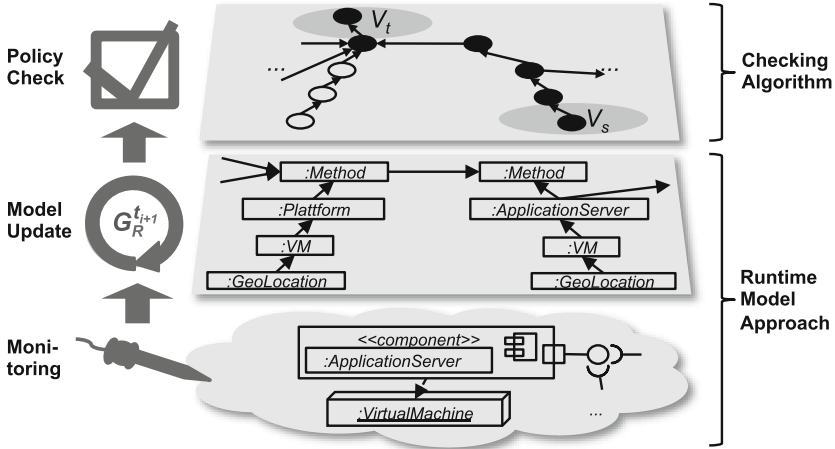


Fig. 1. Overview of R-PRIS

This paper focuses on two main new contributions introduced by the extended R-PRIS approach: the use of a typed runtime model to consider data classification changes (Sect. 3.1) and the enhanced privacy policy checks based on this model (Sect. 3.2).

3.1 Typed Runtime Model

The typed runtime model employed for taking data classification changes into account is an extension of the architectural runtime model that we introduced in [16, 17]. The architectural runtime model $G_R^{t_i}$ reflects the deployment of storage and compute nodes (properties 1 and 2 in Table 1), as well as node interactions (property 3 in Table 1) at time point t_i .

In R-PRIS, the updates of $G_R^{t_i}$ are specified as graph transformation rules. Event-condition-action patterns are employed to reason on observed monitoring information as well as to trigger and parametrize the matching graph transformation rules. This results in an updated runtime model $G_R^{t_i+1}$ for time t_{i+1} .

When extending this architectural runtime model with the data classification information (property 4 in Table 1), we implemented a clear separation of concerns. We separated the dynamically evolving information about node

deployments and interactions (architectural runtime model) from the rather stable information about data classifications (data classification model). For the mapping of elements between these two models (e.g., for mapping a *Method* node to its data classification) we utilize the typed graph concept.

The edges and vertices of a typed graph are assigned to types stored in a separate type graph (for an introduction to typed graphs see [4]). We consider the runtime model as the typed graph and the data classifications as the type graph. Similar to the dynamic type checking in type systems, in our work a set of rules defines how typed vertices are to be treated during runtime.

The concept of the typed runtime model will be described more precisely in the following. We use the typed runtime model shown in Fig. 2 as illustrative example. First, we formally define an R-PRIS architectural runtime model as:

Definition 1. Let $G_R^{t_i} = (V_R, E_R, s_R, t_R)$ be a directed graph that models a big data cloud service at a certain point in time t_i . V_R are the vertices, i.e. service entities, and E_R the edges, i.e. relations between the cloud service entities. Function $s_R : E \rightarrow V$ specifies the source vertex of edges E_R and function $t_R : E \rightarrow V$ specifies the target vertex of edges E_R .

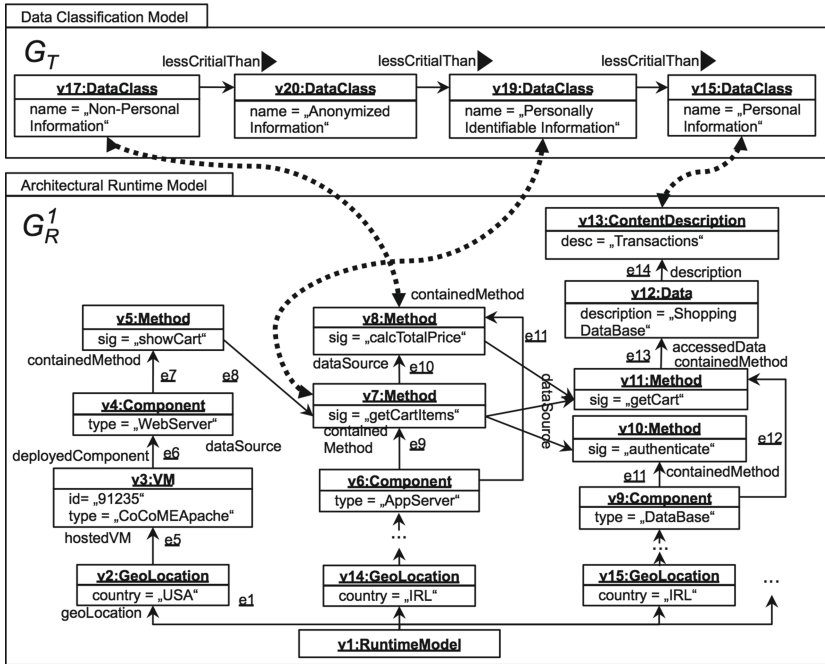


Fig. 2. Example of a typed runtime model in R-PRIS

The lower part of Fig. 2 shows an example of an R-PRIS architectural runtime model, where vertices are expressed as UML objects and edges are expressed as

UML associations. The different kinds of objects and associations are defined by a meta model (introduced in [16,17]). The meta model provides concepts for modeling virtual machines, components, and methods to reflect changes of the compute node property. The deployment relationship reflects changes of geolocations and the data source relationships models changes in interactions.

To express how different data classifications relate to each other in terms of identifiability or sensitivity, we define a data classification model as follows:

Definition 2. Let $G_T = (V_T, E_T, s_T, t_T)$ be the directed graph that represents the data classifications. V_T are the vertices, i.e. data classifications, and E_T the edges, expressing the relations between the data classifications. $s_T : E \rightarrow V$ defines the source vertex of edges $\in E_T$ and $t_T : E \rightarrow V$ defines the target vertex of edges $\in E_T$. Relations between data classifications are transitive⁴, i.e., $(x \succ y) \wedge (y \succ z) \rightarrow (x \succ z)$.

The upper part of Fig. 2 shows an example for a data classification model G_T . It includes the data classes *Personal Information*, *Personally Identifiable Information*, etc. The relationship *lessCriticalThan* reflects the criticality relationship among data classifications.

To interrelate G_R and G_T we exploit the concept of typed graphs. Typed graphs employ graph morphisms for interrelating specific elements of two graphs. We define the graph morphism for R-PRIS as follows:

Definition 3. Let $m : G_R^{t_i} \rightarrow G_T$ be a graph morphism that maps $G_R^{t_i}$ to G_T . Function m is defined as $m = (m_V, m_E)$ with functions $m_V : V_R \rightarrow V_T$ and $m_E : E_R \rightarrow V_T \times V_T$.

In our approach, m_V and m_E are specified as follows. m_V maps *Content-Description* and *Method* entities to data classes V_T of the data classification model. The mapping from *ContentDescription* describes the ‘static’ classification of contents (data objects). The mapping from *Method* describes a change of data classification imposed by executing the method (data processing tasks). All other $G_R^{t_i}$ entities are initially mapped to ‘no’ data classification (\perp). Starting from this initial mapping, we will dynamically update the graph morphism during the policy check in order to determine policy violations (see Sect. 3.2).

Based on Definitions 1–3, we can now define the R-PRIS typed runtime model:

Definition 4. The R-PRIS typed runtime model is defined as $G_{typed} = (G_R^{t_i}, G_T, m)$ with $G_R^{t_i}$ as the architectural runtime model instance, G_T the data classification model, and m the graph morphism that maps $G_R^{t_i}$ to G_T .

When processing data, methods can change the classifications of the processed data. For instance, method *getCartItems* (*v8* in Fig. 2) changes personal information to personally identifiable information. To reflect this, we define data classification changes as:

⁴ Which implies that G_T has to be acyclic.

Definition 5. Given a typed runtime model G_{typed} , let $v_k \in G_R^{t_i}$ represent a method or a data object and let $v_l \in G_R^{t_i}$ represent a method that accesses data provided by v_k . v_l either retains the data classification of the accessed data, i.e. $m(v_k) = m(v_l)$ or changes the classification to a less critical one, i.e. $m(v_k) \succ m(v_l)$. We define the change (in the latter case) as data classification change.

3.2 Privacy Policy Checks

The main idea of our privacy policy check is to express the check as a reachability analysis on the architectural runtime models. In [16] we have formalized and realized this reachability analysis as an st-connectivity problem on the architectural runtime model, thereby considering node deployment and interaction but not data classification changes.

3.2.1 Data-Classification-Aware Search

In order to cover data classification changes in big data cloud services, we extend the reachability analysis by the data classification model, i.e., we search the typed runtime model (including the type model, see Sect. 3.1). Let's take a policy $p = (geo, class)$, which prescribes that data classified as *class* must neither be processed nor stored at the specified geo-location *geo*. The reachability analysis then aims to find a 'violation path' that connects the 'forbidden' geo-location vertex for *geo* with the data classification vertex for *class*. In simple terms, a 'violation path' is a sequence of vertices connected by edges that do not exhibit a data classification less critical than *class*. If such a path exists, the cloud service either stores or processes data at the forbidden geo-location, thereby violating the checked policy p .

As an example, let's define $geo = USA$ and $class = PersonalInformation$, which means that our cloud service may not store or process personal information in the US. Using the typed runtime model of the cloud service in Fig. 2, the reachability analysis aims to find a 'violation path' that connects the 'forbidden' geo-location vertex *USA* (v_2) with the data classification vertex *PersonalInformation* (v_{15}). As can be seen, such a path cannot be found, because although v_{13} is typed as *PersonalInformation*, the path from v_{13} to v_2 traverses vertices which change the data classification to less critical levels; e.g., v_7 changes it to *PersonallyIdentifiableInformation*. This means that the example cloud service (in the current deployment reflected in the typed runtime model) complies with the privacy policy.

To realize the described reachability analysis, the policy check starts with defining $v^{geo} \in G_R^{t_i}$ as start vertex and $v^{class} \in G_R^{t_i}$ as target vertex (with the geo-location and the data classification specified in the policy to be checked). The check then performs a depth first search of the typed runtime model, during which three main mechanisms are employed:

- **Early Termination:** The search terminates the traversal of the current path as soon as it reaches a vertex v_n classified as less critical than *class*. Regardless

of the data classifications along any continuation of the current path, data that passes through v_n will always be less critical than what is expressed in the privacy policy to be checked. This does not exclude the existence of other paths from v^{geo} to v^{class} . Thus, we terminate the search at v_n and then backtrack to v_{n-1} to explore other paths to reach v^{class} .

- **Backtracking:** To facilitate backtracking, and thus not have to start from v^{geo} each time we have terminated a path traversal, we employ the graph morphisms introduced in Sect. 3.1 to store the data classification for all vertices we have traversed thus far. This allows us to continue from v_{n-1} by retrieving the classification we have computed at a certain point of traversal.
- **Classification Traversal:** The traversal of $G_R^{t_i}$ is continued as long as the classification of the successor vertex v_{n+1} is higher or the same as the classification of the current vertex v_n . The continuation is decided by expanding the search space to G_T (also in the case of backtracking). If there exists a path from the data classification of v_n to the data classification of v_{n+1} , then the search continues. In this case, the method returns data equally or more critical⁵ than specified in p .

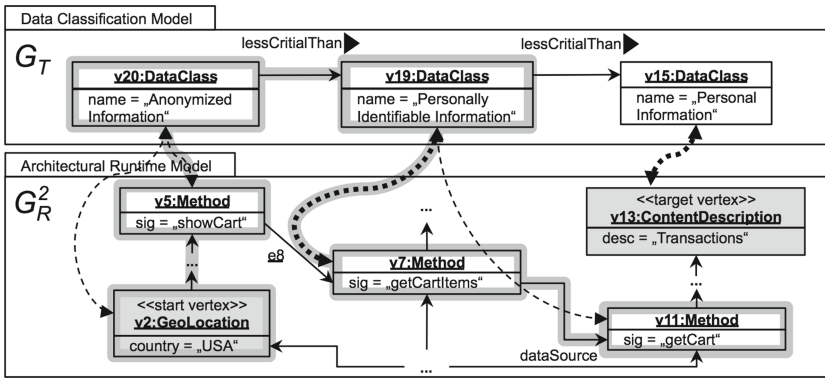


Fig. 3. Example traversal through the typed runtime model

3.2.2 Performance Optimization

When traversing $G_R^{t_i}$ and dynamically classifying $V_R \in G_R^{t_i}$ there is a situation that requires the utilization of a search heuristic for avoiding the re-visiting of nodes. This is important as revisiting nodes would negatively impact on the approach’s time complexity.

Let us assume the check passes a vertex v_s classified with $m_V(v_s) = c_1$. Further, the traversal leads over two crossing paths to a vertex v_t with $m_V(v_t) = c_1$. One of these paths includes a v_1 with $m_V(v_1) = c_1$ and the other path $m_V(v_2) = c_2$ (with $(c_1, c_2) : lessCriticalThan$). If the path over c_2 is traversed

⁵ A policy concerning less critical data must also hold for more critical data.

first, then this traversal results in a false negative (the actual violation is not detected). The reasons are (i) that c_1 is less critical than c_2 such that the checked policy holds and (ii) the path over c_1 to v_1 is not traversed without visiting nodes twice, which blocks the traversal of the path for c_1 . In order to tackle this issue without re-visiting nodes, the check visits vertices adjacent to a vertex v with respect to their criticality in an ascending order (by omitting classifications less critical than $m_V(v)$), which avoids the blocking effect.

3.2.3 Example Policy Check

Figure 3 shows an excerpt of G_R^2 , which is an update of G_R^1 . It shows the morphism-based typings to ease backtracking as thinner arrows, whereas the broader arrows are the initial mappings.

Let us assume, we want to check policy $p = (USA, AnonymizedInformation)$. We dynamically type the start node with the classification specified in p , i.e. $v2 \mapsto v20$. The algorithm starts to traverse G_R^2 and dynamically resolves \perp with the classifications of the preceding vertices (enabling backtracking). The classification of $v5$ is set to $v20$. The classifications of $v5$ and $v7$ differ. Thus, the search is expanded to G_T (classification traversal). As the classification of $v7$ is reachable from the classification of $v5$ the search continues. $v11$ is dynamically typed with the classification of the preceding note, i.e. $v11 \mapsto v19$. After traversing the data vertex $v12$ and typing it with *PersonallyIdentifiableInformation* the algorithm visits $v13$ (not shown in the figure). The algorithm checks whether there is a path from *PersonallyIdentifiableInformation* to the classification of $v13$ in G_T , which is the case. In consequence, data that is classified more critical than v^{class} can be transferred into the USA, which violates p .

4 Proof of Concept Implementation

To demonstrate the feasibility and applicability of R-PRIS, we developed a prototype implementation that we deployed on actual cloud infrastructures. This R-PRIS prototype is also used during our experimental evaluation in Sect. 5.

4.1 Prototype Architecture

The R-PRIS prototype consists of six main components (see Fig. 4): the monitoring probes, the monitoring server, the event processor, the model controller, the policy checker, and a third-party host geo-location service. The probes are deployed on the virtual machines that host the big data cloud service’s software components. The monitoring server forwards the parsed monitoring information to the event processor. The event processor invokes a REST service⁶ for resolving the VM geo-location. The processor triggers the model controller to execute model transformation rules that modify the runtime model with respect to the observed service changes. We use the Henshin graph transformation API for performing the model updates. Henshin supports runtime model based on Ecore⁷.

⁶ <http://freegeoip.net/json/>.

⁷ <http://www.eclipse.org/modeling/emf/>.

For more details on this model-update approach, see [17]. After updating the model, the model controller triggers the policy checker component, which checks the runtime model against the privacy policies (see Sect. 3.2).

The components we developed have been implemented in Java SE 1.7. They are deployed on a dedicated server hosted at our institute (R-PRIS server in Fig. 4) to have maximum control over the prototype and facilitate performance measurements without external influences. The server is equipped with 4 GB of RAM and a single core 2 GHz processor.

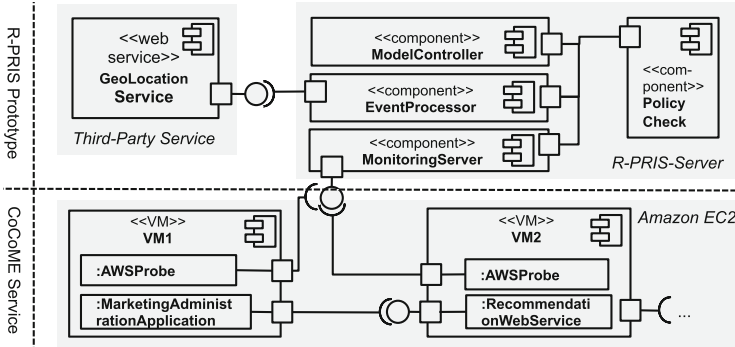


Fig. 4. R-PRIS prototype and parts of the example service

4.2 Cloud Service and Infrastructure

To test the applicability of R-PRIS, we employ a realistic cloud service that builds on the CoCoME case study [14]. CoCoME represents a typical trading service operated by a supermarket chain. In the CoCoME scenario of interest, the CoCoME service sends personalized recommendation e-mails to online customers by exploiting a Hadoop cluster for big data analytics. The Amazon reference architecture for e-commerce websites⁸ is used as the underlying structure for the CoCoME service components. The architecture includes a recommendation web service, a marketing administration application, a Hadoop name node, and Hadoop data nodes. All virtual machines are instrumented with the R-PRIS monitoring probes. The lower half of Fig. 4 shows a subset of these virtual machines and the CoCoME components they host. We choose Amazon EC2 as a realistic execution environment for the cloud services.

4.3 Change Scenarios

To assess the applicability of R-PRIS, we expose the prototype to four change scenarios. These scenarios cover all changes identified in Sect. 2. For each change,

⁸ <http://aws.amazon.com/architecture/>.

we have defined a scenario with positive (policy violation) and negative (policy compliance) situations.

Figure 5 shows excerpts of the actual runtime models (as Eclipse EMF trees). As a pre-condition for each scenario, the virtual machines of the CoCoME service are deployed on a data center in Ireland (see $G_R^{InitConf}$ in Fig. 5). The other models show situations after executing the change scenarios. In each change scenario, one virtual machine is migrated to the US and thus potentially provokes a policy violation. Employing migration is sufficient in our change scenarios, as from a technical perspective, a migration shuts down a virtual machine at the source location and re-starts it at the target location (which leads to initializing a new node, new interactions, etc.; cf. [1]).

Table 2 shows the covered properties, the provoked violation, the applied changes, and the runtime model updates as well as the policy checker results. As the results from the change scenarios indicate, R-PRIS is able to keep the typed runtime model in sync with the reflected CoCoME service. Moreover, the policy checks correctly determine violations and compliance to CoCoME's privacy policy. In particular, the correct true negative check after the migration of the recommendation web service (case G_R^{CS2}) demonstrates that R-PRIS is able to avoid false positives that stem from ignoring data classification changes.

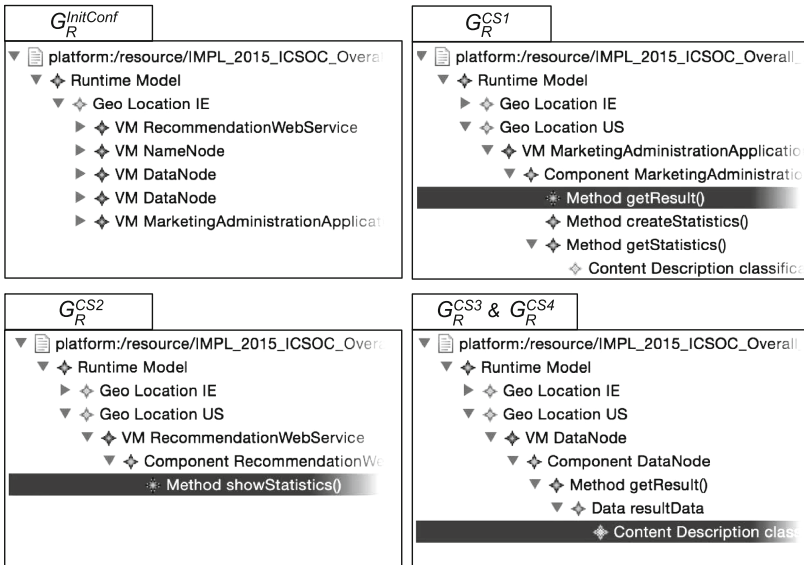


Fig. 5. Screenshots of runtime models (Eclipse EMF) generated by R-PRIS

Table 2. Executed change scenarios and observations

Covered property	Scenario	Migrated VM	Observed R-PRIS Behaviour
Compute node geo-location (1), node interactions (3), data classification (4)	Positive	Marketing administration application	The runtime model reflects the applied change correctly (see G_R^{CS1} in Fig. 5). The check detects a policy violation. The check message includes the violating data flow of personal data that starts in one of the <i>DataNodes</i> , leads over the <i>NameNode</i> and is requested by the <i>getResult()</i> method (marked grey in the figure).
	Negative	Recommendation web service	The runtime model reflects the change correctly (see G_R^{CS2} in Fig. 5). The check assesses the CoCoME service as policy compliant. The <i>showStatistics()</i> method of the recommendation web service invokes the <i>getStatistics()</i> method of the marketing administration application. <i>getStatistics()</i> accesses personal information from the data nodes, but changes the data's classification from <i>PersonalInformation</i> to <i>PersonallyIndentifiableInformtion</i> . The changed classification does not violate the privacy policy. By reflecting the classification change, the check avoids a false positive in comparison to a check that does not take data classification into account.
Storage geo-location (2), interactions (3), classifications (4)	Positive	Data node (personal data)	The runtime model reflects the change correctly (see G_R^{CS3} & G_R^{CS4} in Fig. 5). The check detects the policy violation. Storing personal data at excluded geo-locations violates the privacy policy.
	Negative	Data node (non-personal data)	The runtime model reflects the change correctly (see G_R^{CS3} & G_R^{CS4} in Fig. 5). The check assesses the CoCoME service to be policy compliant. Only non-personal information is stored in the US.

5 Performance Evaluation

Policy violations need to be detected timely in order to have sufficient time to mitigate and respond to these violations. Thus, as one key criterion of R-PRIS we evaluate its performance. On the one hand, we measure its response time based on the aforementioned change scenarios, thereby determining values for realistic application scenarios (Sect. 5.1). On the other hand, we analyze the runtime complexity by means of the \mathcal{O} -notation in order to determine the scalability of the approach (Sect. 5.2).

5.1 Experimental Evaluation

We evaluate the performance of R-PRIS by taking dedicated measurements for its three main phases: (phase 1) cloud monitoring, including sending monitoring

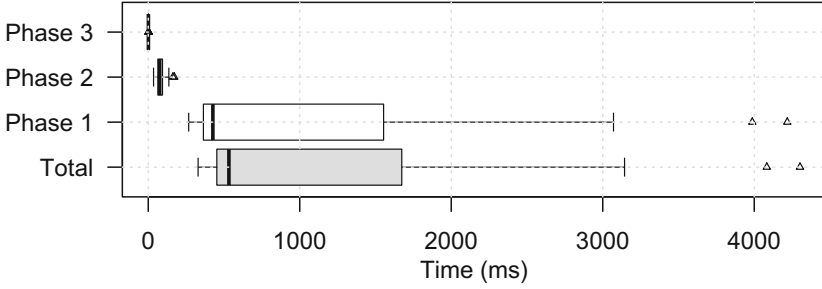


Fig. 6. Response times per phase (on Amazon Web Services)

Table 3. Measurement per phase and total

Phase	Average (ms)	Median (ms)	Minimum (ms)	Maximum (ms)
3	1	1	1	3
2	86	76	35	172
1	113	462	267	4218
Total	1119	532	329	4032

data to the R-PRIS server and resolving host geo-location using third-party service; (phase 2) runtime model update; and (phase 3) policy check.

We use the prototypical implementation and the change scenarios introduced in Sect. 4. The scenarios include violations (i.e., best and typical cases for reachability analysis) and non-violations (i.e., worst cases for reachability analysis as the entire graph has to be traversed). During the execution we repeated the four change scenarios five times each. Thus, we measure 20 response times for each phase during the experiment. The results are shown in Fig. 6 and in Table 3.

We consider the measured worst case response time of around 4 seconds promising. However, in order to speed up the response further, we analyzed the time consumption in phase 1 as this phase predominates the overall duration.

Further investigation showed that the comparatively high response times as well as the outliers in phase 1 stem from employing a third-party service for resolving the host-geolocation. Thus, we choose an alternative cloud infrastructure that offers ‘built-in’ geo-location APIs. We have repeated the measurements for phase 1 on the Azure cloud⁹, which offers a REST API for geo-locating virtual machines. The measured results for phase 1 on Azure are: $avg = 488$ ms, $med = 441$ ms, and $max = 924$ ms. This shows a clear reduction of the overall worst case response time, but exhibit low impact on the med value. The reason might be that the quality of the built-in geo-location API is generally more stable than the third-party service.

⁹ <http://azure.microsoft.com>.

A further performance improvement may be possible if cloud management APIs were available that emit monitoring events as soon as cloud migrations are triggered. In this case, the time that it takes for performing cloud migration or starting a new virtual machine (which may well be in the order of tens or hundreds of seconds [11]), may be used to run the policy checks and stop the migration if it turns out to violate the policy.

5.2 Runtime Complexity Analysis

The above experiments delivered concrete response times for a cloud service that was deployed on a small cloud cluster. To assess the scalability of R-PRIS with respect to performance, we perform a complexity analysis of the approach.

The worst case runtime complexity for depth first search is given by $\mathcal{O}(|V| + |E|)$. As the approach visits the vertices of V_R once at most (see Sect. 3.2.2), the worst case complexity of traversing G_R^{ti} is $\mathcal{O}(|V_R| + |E_R|)$. However, in our approach we perform two interwoven depth first searches. For every pair of adjacent vertices the typed graph is traversed, such that $|E_R|$ resolves to $\mathcal{O}(|E_R| \cdot (|V_T| + |E_T|))$. The overall worst case complexity of the policy check is thus given by $f(G_{typed}) \in \mathcal{O}(|V_R| + |E_R| \cdot (|V_T| + |E_T|))$.

The actual complexity of $f(G_{typed})$ is quite low, when taking knowledge about the application context into account. The data classifications and their relations represented in G_T are derived from standards. Thus, $|V_T|$ and $|E_T|$ have low values (in the order of 10) and, more importantly, are considered to be constant (in case of FIPS, $|V_T| = 3$ and $|E_T| = 2$). According to the \mathcal{O} simplification rules, constants are to be neglected when analyzing worst case complexity. Thus, the checking function's complexity reduces to $f(G_{typed}) \in \mathcal{O}(|V_R| + |E_R|)$ and scales linearly with the size of the runtime model.

6 Related Work

In this section we discuss how R-PRIS relates to existing privacy checks of cloud services and existing runtime model approaches.

Privacy Policy Checks: Research on privacy policy compliance of big data and cloud services mainly focusses on policy violation prevention and compliance monitoring. In policy violation prevention, privacy-by-design principles guide the design and implementations of privacy aware architectures. For instance, the approach presented in [6] equips cloud services with mechanisms that permit or grant data access after matching the client characteristics with privacy policies. However, changes of data geo-locations imposed by migration or replication of the component storing the data are not considered. Data transfers between the client services and further services are not covered. Transitive data transfers that may lead to policy violations thus remain undetected.

Compliance monitoring approaches such as [7, 10] employ cloud services audits during runtime. For instance, the approach in [7] correlates ping round-trip

times of the audited service with geographical information. This allows to determine whether the service interface resides at specific geo-locations. However, the software components behind the service interfaces might be migrated or replicated, while the service interface remains at the same geo-location. For instance, Hadoop data nodes might be replicated to different locations while the request handling master node remains invariant. Further, policy checks may exploit elasticity events [1], by checking elasticity events against policies. Although this would enable local checks, it would not cover the analysis of data classification changes across software components.

Runtime Models: Runtime models provide global views on cloud services. Behavioral runtime models utilize, e.g., sequence-models [12], workflow models [9, 15], and Markov-chains [5]. These models include activities and interactions of the reflected applications but do not provide information about computing nodes, their geo-locations, and the processed data. In contrast, architectural runtime models, e.g., [2, 8], combine behavioral aspects of the system with structural information. These models do not provide information on the geo-location, processed data, and changes of data classifications. However, our runtime model approach [17] provides the required information as being designed for supplying the policy check with the necessary reflections of real world systems.

7 Conclusion and Future Work

We addressed the challenges involved in checking the compliance of big data cloud services against data geo-location policies. In particular, we have addressed the problem of considering different data classifications in order to avoid false positive violations. The main ideas underlying our approach were (1) using typed runtime models that reflect cloud services and data classifications, as well as (2) exploiting efficient reachability analyses on these runtime models to detect policy violations. Our proof of concept implementation and the experimental evidence indicates that the proposed approach is able to correctly identify policy violations with reasonably fast response times. In future work, we plan to investigate the applicability of R-PRIS to a real life example. Further, we plan to complement the approach with pro-active policy violation detection. To this end, we envision the assessments of adaptation plans (e.g., expressed in terms of prescriptive runtime models) before their execution.

Acknowledgements. This work was partially supported by the DFG (German Res. Found.) under Priority Programme “SPP1593” (grant PO 607/3-1).

References

1. Aceto, G., Botta, A., de Donato, W., Pescapè, A.: Cloud monitoring: A survey. *Comput. Netw.* **57**(9), 2093–2115 (2013). <http://www.sciencedirect.com/science/article/pii/S1389128613001084>

2. Brosig, F., Huber, N., Kounev, S.: Automated extraction of architecture-level performance models of distributed component-based systems. In: 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE) (2011)
3. Chen, H., Chiang, R.H., Storey, V.C.: Business intelligence and analytics: From big data to big impact. *MIS Q.* **36**(4), 1165–1188 (2012)
4. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag New York Inc., Secaucus (2006)
5. Epifani, I., Ghezzi, C., Mirandola, R., Tamburrelli, G.: Model evolution by run-time parameter adaptation. In: 31st International Conference on Software Engineering (ICSE) (2009)
6. e Ghazia, U., Masood, R., Shibli, M.: Comparative analysis of access control systems on cloud. In: 2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel Distributed Computing (SNPD) (2012)
7. Gondree, M., Peterson, Z.N.: Geolocation of data in the cloud. In: Proceedings of the third ACM Conference on Data and Application Security and Privacy, CODASPY 2013. ACM, New York (2013)
8. Huber, N., Brosig, F., Kounev, S.: Modeling dynamic virtualized resource landscapes. In: Proceedings of the 8th International ACM SIGSOFT Conference on Quality of Software Architectures (2012)
9. Ivanović, D., Carro, M., Hermenegildo, M.: Constraint-based runtime prediction of SLA violations in service orchestrations. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) *Service Oriented Computing*. LNCS, vol. 7084, pp. 62–76. Springer, Heidelberg (2011)
10. Juels, A., Oprea, A.: New approaches to security and availability for cloud data. *Commun. ACM* **56**(2), 64–73 (2013)
11. Mao, M., Humphrey, M.: A performance study on the VM startup time in the cloud. In: 2012 IEEE 5th International Conference on Cloud Computing (CLOUD), pp. 423–430
12. Maoz, S.: Using model-based traces as runtime models. *Computer* **42**(10), 28–36 (2009)
13. Park, S., Chung, S.: Privacy-preserving attribute distribution mechanism for access control in a grid. In: 21st International Conference on Tools with Artificial Intelligence (2009)
14. Rausch, A., Reussner, R., Mirandola, R., Plasil, F. (eds.): *The Common Component Modelling Example (CoCoME)*. LNCS, vol. 5153. Springer, Heidelberg (2011)
15. Schmieders, E., Metzger, A.: Preventing performance violations of service compositions using assumption-based run-time verification. In: Abramowicz, W., Llorente, I.M., Surridge, M., Zisman, A., Vayssière, J. (eds.) *ServiceWave 2011*. LNCS, vol. 6994, pp. 194–205. Springer, Heidelberg (2011)
16. Schmieders, E., Metzger, A., Pohl, K.: A runtime model approach for data geolocation checks of cloud services. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) *ICSOC 2014*. LNCS, vol. 8831, pp. 306–320. Springer, Heidelberg (2014)
17. Schmieders, E., Metzger, A., Pohl, K.: Architectural runtime models for privacy checks of cloud applications. In: Proceedings of the 7th International Workshop on Principles of Engineering Service-Oriented and Cloud Systems, PESOS 2015, ACM, New York (2015)
18. Szvetits, M., Zdun, U.: Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime. *Softw. Syst. Model.*, Dec 2013

Optimizing Workload Category for Adaptive Workload Prediction in Service Clouds

Chunhong Liu¹(✉), Yanlei Shang¹, Li Duan^{1,2}, Shiping Chen²,
Chuanchang Liu¹, and Junliang Chen¹

¹ State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing 100876, China
{liuchunhong2012, shangyl, duanli, lcc3265, chjl}@bupt.edu.cn

² DATA61, Commonwealth Scientific
and Industrial Research Organization, Sydney, Australia
Shiping.Chen@csiro.au

Abstract. It is important to predict the total workload for facilitating auto scaling resource management in service cloud platforms. Currently, most prediction methods use a single prediction model to predict workloads. However, they cannot get satisfactory prediction performance due to varying workload patterns in service clouds. In this paper, we propose a novel prediction approach, which categorizes the workloads and assigns different prediction models according to the workload features. The key idea is that we convert workload classification into a 0–1 programming problem. We formulate an optimization problem to maximize prediction precision, and then present an optimization algorithm. We use real traces of typical online services to evaluate prediction method accuracy. The experimental results indicate that the optimizing workload category is effective and proposed prediction method outperforms single ones especially in terms of the platform cumulative absolute prediction error. Further, the uniformity of prediction error is also improved.

Keywords: Cloud computing · Resource provisioning · Workload prediction · 0–1 programming

1 Introduction

Service clouds are a kind of service platform using cloud computing technology, on which lots of application systems are running. These applications are designed according to Service-Oriented Architecture (SOA), and it encapsulates business processes into services. Adoption of the new platform paradigm by service provider could make energy and cost to reduce obviously due to the resource provisioning advantages of cloud computing.

With the ability of dynamic provisioning resources, cloud computing platform can adjust the resource to meet the demand of application, which is an important characteristic and is different from traditional platform. By using auto scaling technology, service clouds provide on-demand resource to users. Service providers

are free from considering over-provisioning or under-provisioning for a service. In order to enable the scalability, it is necessary to develop a mechanism to scale up or down virtual machine (VM) automatically. Many cloud providers, such as AWS EC2 [1] and Google the App Engine [2], focus on the scalability, that means providing on-demand computing power and storage capacities dynamically. In these cases, cloud computing is a good choice to liberate service providers from deploying physical infrastructures.

In order to take advantage of scalability, service clouds need to support auto scaling technology in a way that service clouds manage VM instances automatically [3]. Therefore, it is of important significance to realize the resources redistribution via predicting workload. Workload prediction is a key step to realize redistribution and improve the accuracy of distribution in service clouds.

Currently, most workload prediction methods generally utilize a single prediction method [3–18], which choose special predicting model aiming at a certain workload feature. However, these methods could not get satisfactory prediction performance for service clouds. Because the service is different in service clouds and the service workload changes in a variety forms.

Service workloads have various patterns influenced by service type. As key characteristic of workloads, burstiness and self-similarity [19–21] have been reported for some kinds of application in cloud. There are also research works [22–24] analyze that the actual cloud computing workloads are highly time-varying in nature. Wang [21] classified the workloads into slow time-scale data and fast time-scale data according to the speed of load change in form of time-series employing. They described the relationship of two kinds of workload and characterized the impact of the workload on the value of dynamic resizing.

Focus on the variation pattern of workloads, a prediction method based on feature discriminations could be used to obtain a more accurate prediction effect in service clouds. In the project, the common workload classification method is to extract the feature of workload, classify the workload by comparing the feature value with a threshold value. Therefore, the determination of threshold value greatly influence the workload category. However, the workloads are dynamically changing in service clouds. The threshold need dynamic change with the service numbers floating on account of the service adding or reducing at any time in service clouds. However, dynamic adjustment of the threshold needs a lot of historical statistics or experience, and this increases the computation and management difficulty. The key problem of workload classified prediction is how to catalog workload effectively, which is the crux of the matter to obtain more accurate total prediction results.

In this paper, we propose a prediction approach based on feature discrimination. The key idea is that we transform the workload classification threshold problem into the task assignment one. By modeling and solving the integer programming, the service is allocated a suitable forecasting method automatically. We formulate an optimization problem to maximize prediction precision. The problem is then converted to a 0–1 programming problem. Our objective is to design an intelligent workload prediction management architecture with more

robust characteristics to adapt various workload change modes and minimize the predicting error of service clouds. Our contributions are summed up as follows:

1. We propose a categorical prediction approach according to different change mode of workload in order to get higher platform prediction accuracy. Our work applies feedback from the latest observed workloads to a prediction model and update the model parameter on the run. Our method has well robustness and adaptive ability.
2. We present the optimal way to classify workloads in order to allocate prediction models adaptively. We establish a 0–1 programming model and use the sum of l_2 -norm of workload average rate to trade off the prediction accuracy and time. The optimizing solution can dynamically determine the type of service workloads effectively.

The rest of the paper is organized as follows. Section 2 describes the design of our adaptive management architecture for workload prediction. The workload classification optimize model is established and solution is given online in Sect. 3. Section 4 shows the evaluation. We present the related work in Sect. 5, and conclude this paper with Sect. 6.

2 System Architecture

We propose the architecture of workload prediction based on feature discrimination in service clouds. In the proposed architecture, adaptive workload management can automatically allocate a prediction model to each service by solving the optimal problem.

Figure 1 shows the architecture of the classified workload prediction in service clouds. There are Admission Controller module, Resource Manager module, Predictor module and Data Warehouse module in the proposed architecture. Infrastructures include compute, storage and network resources. The services

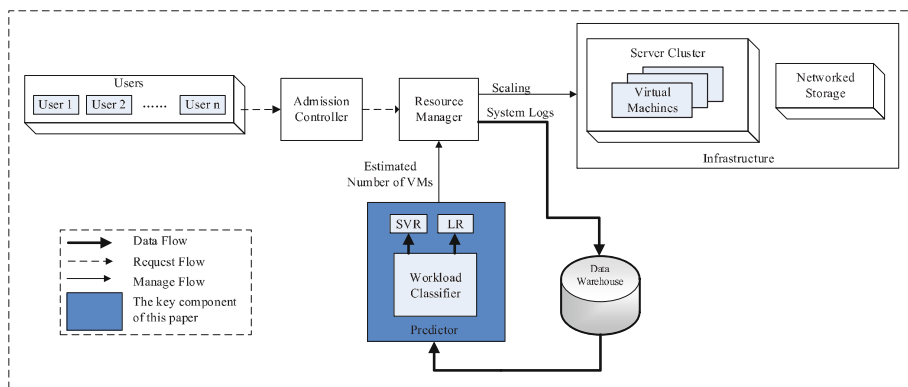


Fig. 1. Adaptive management architecture for workload prediction in service clouds

request is scaled in response to change infrastructures at fixed interval. The fixed interval is denoted as reconfiguration interval. The Admission Controller module determines the user's access. The Resource Manager module adjusts the allocation of resources according to the change of user requests dynamically. All of the service workload logs are transferred to workloads database, named as Data Warehouse. The historical workload information is used to distinguish workload type. It is also used to train and test the prediction model.

The Predictor module is the key module of this paper, it is highlighted in the figure. Once a new service is added, or an old service is down, the Workload Classifier is triggered. Then service workloads could be divided into different data types according to the change pattern of workload. Different prediction method is allocated to appropriate workload based on the optimal solution provided by Workload Classifier. The output of the Predictor is the total workloads of the platform. The total predicting workloads are estimated as the number of VMs and the Resource Manager module takes the number as input. Then the Resource Manager makes the decision to scale up or down VMs for the next reconfiguration interval.

The Workload Classifier automatically allocates suitable prediction model according to the change pattern of each service. In service clouds, there are web service, blog service, audio service, video service, e-mail service and so on. We classify workloads of these services into fast time-scale data or slow time-scale data according to the speed of load change in form of time-series employing. The fast time-scale data is stochasticity and nonlinearity, in which the burstiness of arrivals was high. The slow time-scale data is similar to linearity, in which the peak-to-mean ratio was low. The major difference between fast time-scale data and slow time-scale data is the intensity and speed of change. Therefore, we consider workload average change rate in the workload classification model. The result of workload classification is affected by the behavior of each load recently. Compared with slow time-scale workload, the fast time-scale workload changes violent and its average change rate is larger. This classification well distinguishes the load from the perspective of the change rate.

A constant prediction model cannot cover two kinds of data sets which are with contrary change pattern. In this paper, we utilize linear regression (LR) model and support vector regression (SVR) model to predict the workload, because they are naturally efficient and effective in the forecasting paradigm. Especially SVR is very suitable for the prediction of small sample nonlinear data. Therefore, they are suitable to these change patterns in this paper.

Workload Classifier component is the key to realize adaptive classification prediction. More detailed description can be seen in Sect. 3.

3 Model Formulation and Solution

In this section, the workload category problem is transformed into task assignment one which has a good solution. A 0–1 mixed integer programming of minimum assignment model is established, and an online solution is given in this section.

3.1 The Workload

We consider a discrete-time model that there is a time interval which is evenly divided into “frames” $t \in \{1, \dots, K\}$. In practice, the length of a frame could be 5–10 min. Given a set of time series data $w_t, t \in \{1, \dots, K\}$, $t = 1$ is the first frame of the time series, w_t stands for the actual data in frame t .

3.2 The Workload Classification Problem

Supposed there are i kinds of services $A_i (i = 1, \dots, n)$ in service clouds. There are j kinds of prediction models $M_j (j = 1, 2)$ for A_i , which are fit for forecasting workload according to the characteristic of each kind of application workload provided by service clouds. Assign M_j to A_i , the predicting time is denoted by t_{ij} , the predicting error is denoted by $\varepsilon_{ij} (i = 1, \dots, n, j = 1, 2)$. Here, ε_{ij} expresses Mean Absolute Percentage Error (MAPE) of a service workload i by model j at the frame t , defined as $\varepsilon_{ij} = \frac{1}{n} \sum_{s=1}^n \left| \frac{(\hat{w}_s - w_s)}{w_s} \right|$, where w_s is the actual output, and \hat{w}_s is the prediction output, s is the observing time point, n is the data number of observation dataset at frame t . Cumulative Absolute Error at frame t denoted by $c\varepsilon_t$, and expresses the sum of MAPE of all services loads in service clouds, defined as $c\varepsilon_t = \sum_{i=1}^n \varepsilon_{ti}$.

In order to achieve the minimum $c\varepsilon_t$ and satisfy the time requirement meanwhile, it is the key to improve the platform prediction performance that assigns suitable prediction models to each service in service clouds. While the workload feature is hard to extract and dynamic changing threshold is difficult to determine, we transform that problem into an assignment problem by establishing the optimization model.

Supposed x_{ij} expressed as distributing prediction model M_j to service A_i .

$$x_{ij} = \begin{cases} 0 & A_i \text{ is not assigned prediction model } j \\ 1 & A_i \text{ is assigned prediction model } j \end{cases} \quad (1)$$

When a new service is adding, or an old service is down at frame t , we need to reorganize the service type and reassign the prediction model. Then we need to determine x_{ij} .

3.3 The Workload Classification Optimization

Given the workload classification component above, the platform has one control decision vector x_{ij} , i.e. the allocation of prediction models when service number is changing at each frame. Given the limited error vector and time vector for A_i that satisfy the SLA requirements are $\varepsilon = (\varepsilon_1, \dots, \varepsilon_i, \dots, \varepsilon_n)$ and $T = (T_1, \dots, T_i, \dots, T_n)$. The goal of the service clouds is to determine x_{ij} to minimize the predicting error during $[0, K]$:

$$\begin{aligned}
\min \quad & \sum_{i=1}^n \sum_{j=1}^2 \varepsilon_{ij} x_{ij} \\
\text{s.t.} \quad & 0 < t_{ij} < T_i \quad i=1, \dots, n, \quad j=1, 2 \\
& 0 < \varepsilon_{ij} < E_i \quad i=1, \dots, n, \quad j=1, 2 \\
& \sum_{i=1}^n x_{ij} = 1 \quad j=1, 2 \\
& x_{ij} \in \{0, 1\}^n
\end{aligned} \tag{2}$$

where:

T_i is the upper limit of predicting time for A_i . It equals to configuration time - VM restart time - service deployment time.

ε_{ij} is the predicting error of A_i under current predicting model scheme j , E_i is the maximum error limit of A_i .

This model generalizes the service cloud optimization problem by accounting the total error of the platform. However, there is an issue in the model: the object of minimum platform error leads to the results that the optimal solution tends to be fast time-scale type. If it is assigned as fast data, it will use SVR model to forecast workload. Compared with LR model, SVR model has small predicting error and long predicting time. Predicting time of platform at frame t defined as the maximum predicting time of each service used in frame t . This would lead to more fast items and a long predicting time in the optimal solution for the workload classification.

With regard to this issue, we consider time factor, instead of only considering minimum platform error in the model. There is a significant difference in the average change rate of workload between fast time-scale and slow time-scale. The fast time-scale workload changes violent and the average change rate of workload is larger. Defined workload average change rate as $\bar{\omega} = \sum_{j=1}^n (y_{j+1} - y_j) / N$. There is obviously difference between fast time-scale and slow time-scale in term of $\bar{\omega}$. The $\bar{\omega}$ value of fast type is greater than that of slow type. In a classification result, the more fast type number, the larger sum of l_2 -norm for $\bar{\omega}$. Therefore, to deal with the above issue, the Eq. (2) can be more balanced by introducing the sum of l_2 -norm for $\bar{\omega}$ to tradeoff the overall prediction performance including prediction accuracy and time. Constant λ trades off a priori knowledge as throughout on the influence degree of the model. It needs to be determined before solving the optimization problem. Then the problem (2) may be written as Eq. (3). The influence of λ value on the prediction results will be discussed in Sect. 4.4.

$$\begin{aligned}
\min \quad & \sum_{i=1}^n \sum_{j=1}^2 \varepsilon_{ij} x_{ij} + \lambda \sum_{i=1}^n \sum_{j=1}^2 \|\bar{\omega}_i \cdot x_{ij}\|_2 \\
\text{s.t.} \quad & 0 < t_{ij} < T_i \quad i=1, \dots, n, \quad j=1, 2 \\
& 0 < \varepsilon_{ij} < E_i \quad i=1, \dots, n, \quad j=1, 2 \\
& \sum_{i=1}^n x_{ij} = 1 \quad j=1, 2 \\
& x_{ij} \in \{0, 1\}^n.
\end{aligned} \tag{3}$$

3.4 The Optimal Solution

Equation (3) is a 0–1 integer programming, branch and bound method is a primary algorithm to solve this kind of problem. The classical branch and bound algorithm works for the low efficiency when the scale of the problem is bigger. The branch and sub-problem choice strategies are important factors of the algorithm efficiency [26,27]. We adopt a search approach to solve Eq. (3). In order to achieve the better time performance, we choose the minimum cost priority and feasibility pruning strategies in searching, pruning and bounding process in this section. There is a detailed description in Fig. 2.

Given integrate programming as shown in Eq. (3), $S(P)$ is the feasible solution set of Eq. (3), \bar{x} is the feasible solution, F_u is the upper bound of the optimal value. Programming (3) can be divided into sub-programmings, denoted by $(P_1), (P_2) \cdots (P_i)$, and each sub-programming has corresponding relaxation programming, denoted by $(\bar{P}_1), (\bar{P}_2), \cdots, (\bar{P}_i)$, $S(\bar{P}_i)$ is the feasible solution set of (\bar{P}_i) , $x^{(i)}$ is the optimal solution and f_i is the optimal value. NF is the subscript set of the detecting problem P_i .

All the children nodes are produced from the current node in branch process. Those nodes that are impossible to generate feasible solutions are abandon, and other children nodes are adding to activated nodes set NF . Then new expansion node is chosen from the activated nodes set. In branch process, the minimum $c\varepsilon_k$ is the strategy to choose the branching node. We select the appropriate subscript $k \in \{1, 2, \cdots, n\}$ according to Eq. (4).

$$k = \arg \min \{f(\lfloor x_j^0 \rfloor), f(\lceil x_j^0 \rceil) | x_j^0 \text{ is fraction}\} \quad (4)$$

After k is determined, $x_k = 0$ or $x_k = 1$, and the original problem is divide into two sub-problems.

The boundary of objective function for the sub-problems is established in a bounding process. If a sub-problem value is outside the boundary, this sub-problem will be pruned. To solve the relaxed programming (\bar{P}_i) can obtain the optimal solution $x^{(i)}$ and the optimal value f_i .

In step $i \geq 0$, the low boundary of the optimal value f_i for (P_i) can be obtained, denoted as:

$$LF_{Li} = \min\{f_i | i \in NF\}$$

In step $i \geq 0$, the upper boundary of the optimal value f_i for (P_i) can be obtained, denoted as:

$$UF_{Ui} = \min\{LF(f_i) | f_i \in S(\bar{P})\}$$

To repeat branching, bounding and pruning process until there is no node in NF , the best feasible solution is the optimal solution of the original problem as shown in Eq. (3).

Through the above discussion, the branch and bound algorithm for the prediction model assignment program solving is described as Fig. 2.

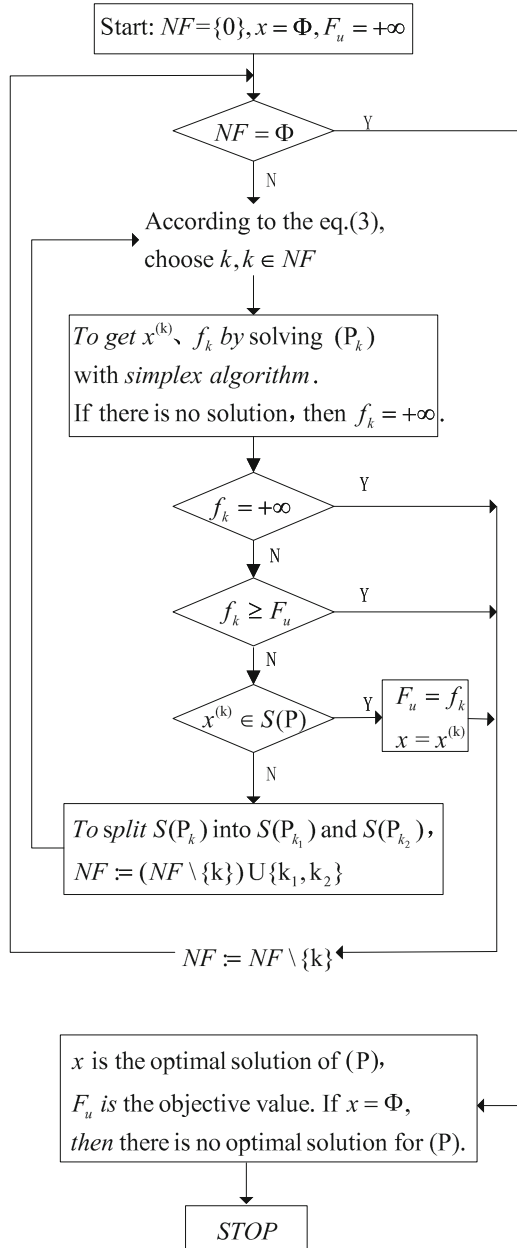


Fig. 2. Flow chart of branch and bound algorithm

4 Experimental Analysis

In order to illustrate the effectiveness of the proposed approach in service clouds, we conduct experiments with the data set built from some typical services of the application system developed by our lab. First, each service is allocated a suitable prediction model according to the optimal solution. With this assignment, we compare our method with single prediction methods in term of the platform cumulative absolute predicting error proposed in this paper. In order to obtain more accurate workload classification by optimal solution, we discuss the effect of parameter λ in optimization model on optimal solution followed.

4.1 Setup of the Experiment

There are some application systems in the experimental service clouds, such as Social Network Sites(SNS), Multimedia Conference System(Video System), Online Learning System(Learning System) and so on. Each of them is composed of more than one services. Table 1 lists some of the services of these systems. They are running in service clouds which are based on OpenStack, using two IBM x3650 servers as control nodes and three IBM x3650 servers as computing nodes.

Table 1. Services of the application system in service clouds

Application	Service
SNS	Web service
	Picture service
	Blog service
	Comment service
	E-mail service
Video system	Video service
	White board service
	Discuss service
	Cache service
Learning system	Web service
	White board service
	Video service
	Discuss service
	E-mail service

The data set of optimal classification experiment is built from the real trace of requests to service servers. We sample workloads of these services every 10 min time interval for 48 h in service clouds. The time interval is set by the time spending on booting a VM. The data set is composed of 30 groups workload traffic samples include 20 groups fast time-scale data plus 10 groups slow time-scale data to evaluate the workload classification optimization result. We would

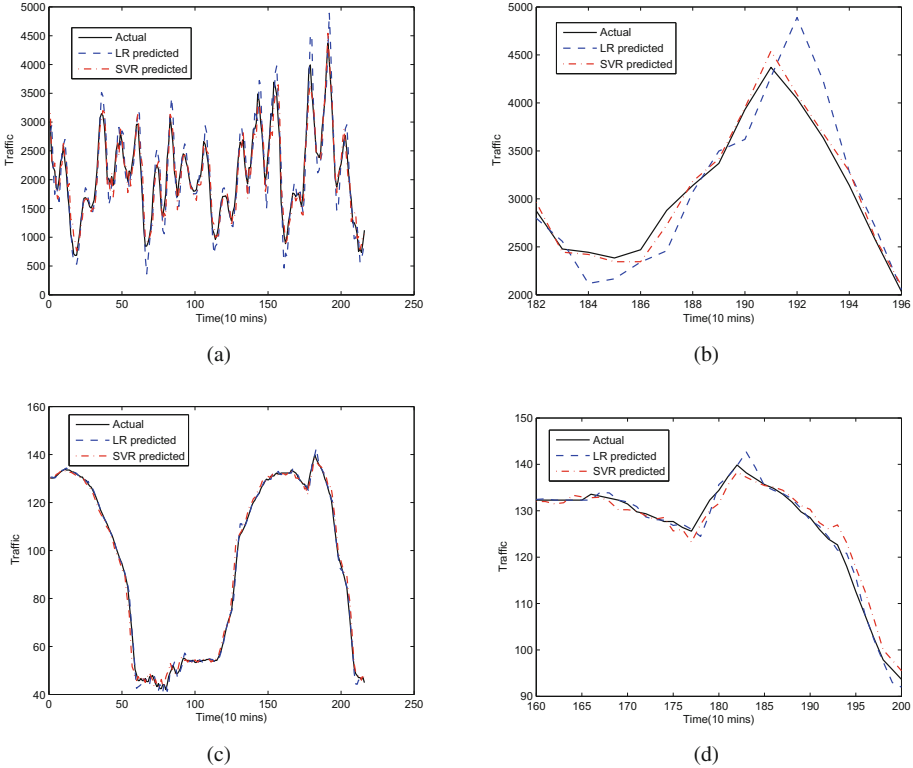


Fig. 3. Comparative prediction results on two kinds of workload: Web Service and Video Service. Figures on the left column illustrate comparative results using LR and SVR algorithm; figures on the right column enlarge the results for more clear illustration. (a) Web Service workload, (b) Enlarged part, (c) Video Service workload, (d) Enlarged part.

conduct our prediction experiment with two typical workloads to evaluate the proposed prediction approach. Workload variation pattern is a stochastic feature, closely related to the characteristics of service applications. For example, the traffic workload of web service is greatly influenced by a web content and change drastically with the arrivals burstiness. Video service mainly provides video transmission and decoding process for multimedia conference system. Its users mainly in the campus tend to use the video service in the daily work time. The traffic curve appears certain regularity and takes the shape of a flat curve. The ratio of peak to average is low and has cyclical change. Then we choose these two service workloads to verify the prediction result.

4.2 Workload Prediction Methods Analysis

The prediction models used in the prediction experiment are LR and SVR. LR [28] models the relationship between one or more input variables z and a dependent output variable y by using a linear equation. We take the following form.

$$y_t = \beta_1 + \beta_2 z_t \quad (5)$$

where y is the target variable, here is prediction workload. z is the explained variable, here is the time. t indexes the sample interval. The coefficients β_1, β_2 are determined by solving a linear regression equation based on previous workloads $y_{t-1}, y_{t-2}, y_{t-3}$ and so on. β_1, β_2 change with different previous workloads, that is to say, this model can change with the workload trend. We use the Ordinary Least Squares to solve the Eq. (5).

Before using the SVR to predict time series, sample space reconstruction is required. We analyze our time series data, and give the reconstruction process as follow. Given time series x_t ($t = 1, 2, \dots, T$), from Takens phase space delay reconstructing theory, we use the m -dimensional vector, it is defined as

$$x(t-1) = [x(t-\tau), x(t-2\tau), \dots, x(t-m\tau)] \quad (6)$$

where m is an embedding dimension and τ is delay constant. The prediction model can be described as $x(t) = g(x(t-1))$, g is a non-linear map. For our time series data, the input variables should be reconstructed using Eq. (6) of all data sets in the beginning, the values of embedding dimension and delay constant for the data set are set as follows: $m = 3, \tau = 1$.

We used the Gaussian RBF kernel and defined as $K(x_i, x_j) = \exp(-\|x_i - x_j\|^2 / (d\delta^2))$, where d is the dimensionality of x , $d = 2$, δ^2 is the variance of the kernel. We used the leave - one out cross validation approach to select model parameters, which chose the test and training sets randomly [29]. This method divided sample data into two parts, 70% of the data is processed as a training set and 30% of the data is processed as a test set. We used LIBSVM toolbox [30] to implement SVR algorithm. SVR parameter Settings are as follows: $m = 3, c = 100, g = 0.5, s = 3, p = 0.001, t = 2$.

Because of adopting cross validation, we use another half of the data set to test the prediction effect. As shown in Fig. 3, both SVR and LR make a good prediction effect for slow time-scale data. For fast time-scale data, the traffic changes drastically, more burstiness of arrivals come out near the sudden change. LR could not adapt the change and it produces a greater predictive error and phase deviation. SVR could adapt the dramatic change tendency well, because it conducts nonlinear kernel transforms. SVR has better prediction effect for fast time-scale data. But SVR needs more predicting time than LR. Therefore, we allocate the linear regression model for slow time-scale data as well as the support vector machine model for fast time-scale data in classified prediction approach.

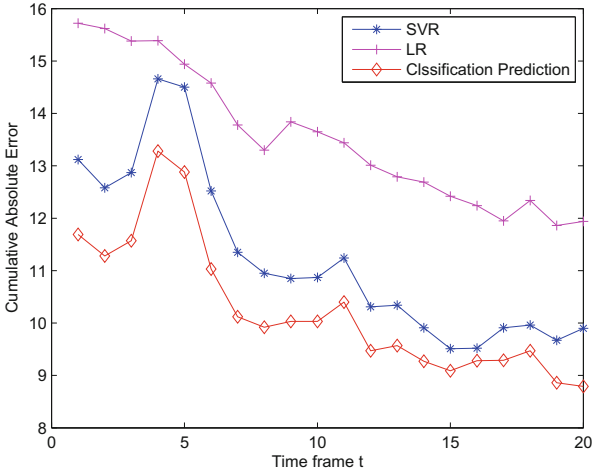


Fig. 4. Comparison of cumulative absolute error

4.3 Classified Prediction Effect Analysis

According to the above prediction model allocation scheme, we compare the prediction effect of classified prediction approach with SVR and LR prediction model. We evaluate the accuracy of the prediction approaches based on a number of metrics: Mean Absolute Percentage Error (MAPE), Cumulative Absolute Error at the frame t ($c\varepsilon_t$).

We select 20 points uniformly within prediction period and compute the MAPE and $c\varepsilon_t$. Then we compare the prediction effect of three approaches in terms of the mean value, the square error, and the mean square deviation of MAPE.

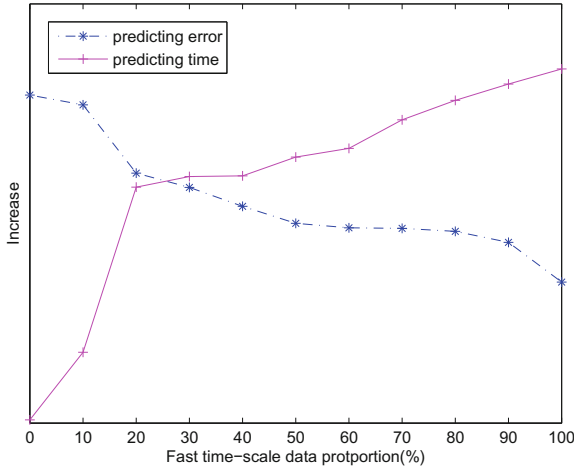
Figure 4 shows $c\varepsilon_t$ of service clouds at 20 prediction points. It illustrates that the MAPE of the classified prediction is minimum, SVR comes second, and LR is the highest.

In regard of the mean value, the square error and the mean square error at 20 prediction points with three methods, the statistical results are listed in Table 2. The mean $c\varepsilon_t$ of three prediction methods (e.g. classified prediction, SVR, LR) are 11.227, 13.544 and 10.266 respectively. Therefore, compared with SVR and LR, the classified prediction method reduces the platform cumulative absolute predicting error by 8.56% and 24.20% respectively. For different service load predictions, the cumulative error of a single prediction method is far bigger than the adaptive classification prediction method depending on the load characteristics.

Table 2 shows the $c\varepsilon_t$ statistical results of service clouds at 20 prediction points. The mean value of the classification prediction method is the smallest, LR is the highest. It illustrates that the $c\varepsilon_t$ of the classification prediction distributes more concentrated and forecasts more accurately. Variance and standard deviation of $c\varepsilon_t$ at each time are consistent. Variance and standard of classifica-

Table 2. Statistical results of cumulative absolute error

	Mean value	Square error	Mean square error
SVR	11.227	2.5997	1.6124
LR	13.544	1.7377	1.3182
Classified prediction	10.266	1.6702	1.2924

**Fig. 5.** Influence of fast time-scale data proportion on error and time.

tion prediction method are the smallest, and that of SVR are the biggest. This illustrates that the $c\varepsilon_t$ of the classification prediction method distribute more stably, compared with single prediction methods.

4.4 Influence of the Parameters λ on Workload Classification Optimization

As analysis in the previous section, the proportion of fast time-scale data in workload classification produces an effect on platform prediction accuracy and platform prediction time in classified predicting approach.

We measure the platform prediction error and platform prediction time by classified prediction approach, which the proportion of fast time-scale data increases from 0 to 100% with 30 groups of load respectively. Figure 5 illustrates the influence of fast time-scale data proportion on accuracy and time. Obviously, with the increase of proportion, the platform prediction error decrease distinctly. Meanwhile, this would increase the platform prediction time. Therefore, finding the optimal ratio of fast time-scale workload is the key to improve the platform prediction result and keeping the balance of the forecasting accuracy and time performance to satisfy the response time of SLA.

In the workload classification optimization model, parameter λ plays the role to control the ratio of fast time-scale data in the solution of optimization. λ trades off a priori knowledge of workload on the influence of the model. It

Table 3. The relation between λ and optimal classification results

λ	Accuracy of classification	Average cumulative absolute error	Average predicting time (ms)
0	73.33 %	3.456	72.026
[0.11,1)	93.33 %	3.47	54.053
$+\infty$	33.33 %	8.028	0.0525

needs to be determined before solving the optimization problem. Thus the value of parameter λ on the optimal solution and predicting accuracy performance is studied in this section.

Table 3 illustrates the effect of λ . Here, accuracy of classification is defined as the correct classification number divided by the total number of load. Average cumulative absolute error equals to $c\varepsilon_t$ divided by the number of services. Average predicting time equals to the sum of predicting time divided by the number of services.

From Table 3, we analyze the effect of λ range where global optimal solution can be found.

- (1) $\lambda = 0$ that is only the error term, no regular item to play a role. In optimal solution X, there are 26 workloads identified as fast time-scaled data, 4 workloads identified as slow time-scaled data. Classification accuracy is 73.33 %.
- (2) As λ value is gradually increasing from 0.11 to 1, regular item plays more important role. In optimal solution X, there are 22 workloads identified as fast time-scaled data, 8 workloads identified as slow time-scaled data. 2 slow time-scaled workloads identified as fast time-scaled data on account of severe changing in the test interval. Classification accuracy is 93.33 %. Compared with (1), workload predicting model scheme, average cumulative absolute error only increase 0.4 % but average predicting time reduce 33.25 % in this scheme.
- (3) $\lambda = +\infty$ that is regular item to play an important role, the error term nearly play no function. In optimal solution X, 30 workloads are identified as slow time-scaled data. Because in this optimal model, the objective is only time performance and the predicting time of slow time-scaled is much better than fast time-scaled. Classification accuracy is only 33.33 %. The model with no information of workload average rate of change leads to poor performance.

5 Related Work

At present, the approaches for workload or resource prediction in cloud can be classified in two categories according to the feature of a prediction method.

The first category prediction methods employed classical prediction models by studying the application characteristic in the cloud platform. To satisfy

upcoming resource demands, Islam et al. [4] used Neural Network (NN) and Linear Regression (LR) to get new prediction-based resource measurement and provisioning strategies. Markov method is also used to predict workload. Studies [5–7] show that web and data center workloads tend to present behavior that can be effectively captured by time series-based models. Regard workload as time series, the Autoregressive Integrated Moving Average (ARIMA) model was applied to estimate the future need of applications. Roy et al. [8] utilized linear prediction method such as the exponential moving average, a second order autoregressive moving average and moving average method, to forecast the time series workloads. Sapankevych et al. [9] provided a survey of time series prediction applications using support vector machines (SVM) approach. It points out that the motivation for using SVMs is the ability of this methodology to accurately forecast time series data when the underlying system processes are typically nonlinear, nonstationary and not defined a priori knowledge. SVMs have also been proven to outperform other non-linear techniques including neural-network based non-linear prediction techniques such as multi-layer perceptrons. The work of [10] shows SVM provides the best prediction model among SVM, NN and LR with the SLA metrics for Response Time and Throughput. John et al. [11] analyzed the cloud workloads and used Markov modeling and Bayesian modeling to predict workload. Mathias et al. [12] developed a Markov framework to model the capacity variability of a service cluster for VM provisioning.

The second category prediction methods tended to propose a new prediction approach according to the feature of workload. They usually adopted the single prediction strategy at the same time to predict all the services resources. As typical ones, Caron et al. [13] proposed a Pattern Match algorithm to predict workload taking advantage of the self-similar feature. The main idea is to find out the historical load data similar to the load in current stage by using the feature of web traffic self-similar. Meanwhile, Gong et al. [14] presented a predictive elastic resource scaling scheme named as PRESS, which monitored CPU usage of VM, and derived signature-driven resource demand prediction by employing a Fast Fourier Transform. For applications with no repeating patterns, a discrete-time Markov chain based method was proposed to obtain state-driven resource demand prediction. Ghorbani et al. [15] introduced a fractal operator to account for the time-varying fractal and bursty properties of the cloud workloads.

However, these methods mentioned above usually adopt the single prediction strategy, that is, at the same time using the same method to predict all the services resources. Although many works were dedicated to resource prediction in clouds, there are few researches related to the application scenarios in which a variety of businesses run on the cloud platform at the same time. Considering the influence of cross correlation between servers' workloads, Khan et al. [16] analyzed the interaction between these workloads from the perspective of multiple time series, and then used hidden Markov chain model to predict the workload. Zhang et al. [17] designed a workload factoring service for proactive workload management. It segregated flash crowd workload from base workload with a data item detection algorithm.

Finally, Kupferman et al. [18] recommended a set of scoring metrics to measure the effectiveness and efficiency of dynamic scaling algorithms in terms of availability and cost. Copil et al. [31] presented a framework for estimating and evaluating cloud service elasticity behavior to estimate the expected elasticity behavior in time. It advise elasticity controllers about cloud service behavior to improve cloud service elasticity. Islam et al. [4] discussed a set of predicting metrics to evaluate the accuracy of the prediction algorithms, including Mean Absolute Percentage Error(MAPE), PRED(25), Root Mean Squared Error(RMSE) and R^2 Prediction Accuracy.

6 Conclusions and Future Work

To facilitate auto scaling resource management in service clouds, one of the crucial technologies is to predict resources in advance. In service clouds, it is difficult to predict accuracy due to the variation pattern of workloads.

In order to solve the above problem, we have presented an adaptive workload prediction approach in this paper. The approach categorizes the workloads and assigns different prediction models according to the speed of workload change. The key idea is that we formulate an optimization problem to maximize prediction precision, then convert workload classification into a 0–1 programming problem. It is solved quickly by employing an improved branch and bound algorithm.

We evaluated its accuracy using real traces from some typical services of the application system developed by our lab. We also analyzed the parameters value and the influence of the fast time-scale data on workload classification optimization. The experiment results demonstrate that our approach predicts more accurately in terms of the platform cumulative absolute predicting error. Moreover, the predicting error is well-distributed.

Our approach is also generic, and it can be used well in most service-cloud scenarios. In future, we plan to implement and evaluate our classification prediction approach for a wide variety of workloads by integrating the existing service platforms of our laboratory. We also intend to introduce more prediction models for these workload patterns. We need to modify the optimal model accordingly. This strategy will facilitate the prediction framework to make business-level SLAs constraint (such as, response time, accuracy performance and cost etc.) for adaptive and optimal resource provisioning in the cloud.

Acknowledgement. The work is supported by National Key Technology Research and Development Program of China (Grant No. 2012BAH94F02); National Natural Science Foundation of China (Grant No. 61132001); National High-tech R&D Program of China (863 Program) under Grant No. 2013AA102301; Project of New Generation Broad band Wireless Network (Grant No. 2014ZX03006003);The National Grand Fundamental Research 973 Program of China (Grant No. 2011CB302506).

References

1. Amazon elastic compute cloud. <http://aws.amazon.com/ec2/>

2. Google app engine. <http://developers.google.com/appengine/>
3. Jingqi, Y., Chuanchang, L., Yanlei, S., et al.: A cost-aware auto-scaling approach using the workload prediction in service clouds. *Inf. Syst. Front.* **16**(1), 7–18 (2014)
4. Islam, S., Keung, J., Lee, K., et al.: Empirical prediction models for adaptive resource provisioning in the cloud. *Future Gener. Comput. Syst.* **28**(1), 155–162 (2012)
5. Calheiros, R.N., Masoumi, E., et al.: Workload prediction using ARIMA model and its impact on cloud applications' QoS. *IEEE Trans. Cloud Comput.* **2**(8), 1–11 (2014)
6. Tran, V.G., Debusschere, V., Bacha, S.: Hourly server workload forecasting up to 168 hours ahead using seasonal ARIMA model. In: *Proceedings of the 13th International Conference on Industrial Technology (ICIT 2012)*, pp. 1127–1131 (2012)
7. Jiang, Y., Perng, C.S., Li, T., et al.: Cloud analytics for capacity planning and instant vm provisioning. *IEEE Trans. Netw. Serv. Manage.* **10**(3), 312–325 (2013)
8. Roy, N., Dubey, A., Gokhale, A.: Efficient autoscaling in the cloud using predictive models for workload forecasting. In: *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing*, pp. 500–507, IEEE (2011)
9. Sapankevych, N.I., Sankar, R.: Time series prediction using support vector machines: a survey. *IEEE Comput. Intell. Mag.* **4**(2), 24–38 (2009)
10. Bankole, A.A., Ajila, S.A.: Cloud client prediction models for cloud resource provisioning in a multitier web application environment. In: *Proceedings of 2013 IEEE 7th International Symposium on Service Oriented System Engineering (SOSE)*, pp. 156–161, IEEE (2013)
11. Panneerselvam, J., Liu, L., Antonopoulos, N., et al.: Workload analysis for the scope of user demand prediction model evaluations in cloud environments. In: *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC)*, pp. 883–889 (2014)
12. Björkqvist, M., Spicuglia, S., Chen, L., Binder, W.: QoS-aware service VM provisioning in clouds: experiences, models, and cost analysis. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) *ICSOC 2013. LNCS*, vol. 8274, pp. 69–83. Springer, Heidelberg (2013)
13. Caron, E., Desprez, F., Muresan, A.: Forecasting for Cloud computing on-demand resources based on pattern matching. Technical report, INRIA, pp. 1–23 (2010)
14. Gong, Z., Gu, X., Wilkes, J.: Press: Predictive elastic resource scaling for cloud systems. In: *2010 International Conference on Network and Service Management (CNSM)*, pp. 9–16, IEEE (2010)
15. Ghorbani, M., Wang, Y., Xue, Y., et al.: Prediction and control of bursty cloud workloads: a fractal framework. In: *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, pp. 12–21, ACM (2014)
16. Khan, A., Yan, X., Tao, S., et al.: Workload characterization and prediction in the cloud: a multiple time series approach. In: *Network Operations and Management Symposium (NOMS)*, 2012, pp. 1287–1294, IEEE (2012)
17. Zhang, H., Jiang, G., Yoshihira, K., et al.: Proactive workload management in hybrid cloud computing. *IEEE Trans. Netw. Serv. Manage.* **11**(1), 7–18 (2014)
18. Kupferman, J., Silverman, J., Jara, P., Browne, J.: Scaling into the cloud. Technical report, University of California, Santa Barbara; CS270 - Advanced Operating Systems (2009). <http://cs.ucsb.edu/~jkupferman/docs/ScalingIntoTheClouds.pdf>

19. Yin Jianwei, L., Xingjian, Z.X.: BURSE: a bursty and self-similar workload generator for cloud computing. *IEEE Trans. Parallel Distrib. Syst.* **26**(3), 668–680 (2015)
20. Eldin, A.A., Rezaie, A., Mehta, A., et al.: How will your workload look like in 6 years? Analyzing Wikimedia’s workload. In: 2014 IEEE International Conference on Cloud Engineering (IC2E), pp. 349–354 (2014)
21. Wang, K., Lin, M., Ciucu, F., et al.: Characterizing the impact of the workload on the value of dynamic resizing in data centers. In: 2013 Proceedings IEEE International Conference on Computer Communications (INFOCOM), pp. 515–519 (2013)
22. Reiss, C., Tumanov, A., Ganger, G.R., et al.: Heterogeneity and dynamicity of clouds at scale: google trace analysis. In: Proceedings of the Third ACM Symposium on Cloud Computing, pp. 7–20, ACM (2012)
23. Di, S., Kondo, D., Cirne, W.: Characterization and comparison of cloud versus grid workloads. In: Proceedings of IEEE International Conference on Cluster Computing, pp. 230–238 (2012)
24. Liu, Z., Cho, S.: Characterizing machines and workloads on a google cluster. In: Proceedings of International Conference on Parallel Processing Workshops, pp. 397–403 (2012)
25. Jorgensen, M.: Experience with the accuracy of software maintenance task effort prediction models. *IEEE Trans. Softw. Eng.* **21**(8), 674–681 (1995)
26. Chan, D.Y., Ku, C.Y., Li, M.C.: A method to improve integer linear programming problem with branch-and-bound procedure. *Appl. Math. Comput.* **179**(2), 484–493 (2006)
27. Mingfang, N.: Li Qi.: a surrogate constraint bounding approach to mixed 0–1 linear programming problems. *J. Syst. Sci. Math. Sci.* **19**(3), 341–347 (1999)
28. Seber, G.A.F., Lee, A.J.: *Linear Regression Analysis*. Wiley, New York (2012)
29. Mao, W., Xu, J.: Cao Xizheng.: a fast and robust model selection algorithm for multi-input multi-output support vector machine. *Neurocomputing* **130**, 10–19 (2014)
30. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. *ACM Trans. Intell. Syst. Technol. (TIST)* **2**(3), 1–27 (2011)
31. Copil, G., Trihinas, D., Truong, H.-L., Moldovan, D., Pallis, G., Dustdar, S., Dikaiakos, M.: ADVISE – a framework for evaluating cloud service elasticity behavior. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) ICSOC 2014. LNCS, vol. 8831, pp. 275–290. Springer, Heidelberg (2014)

On Developing and Operating of Data Elasticity Management Process

Tien-Dung Nguyen, Hong-Linh Truong^(✉), Georgiana Copil, Duc-Hung Le, Daniel Moldovan, and Schahram Dustdar

Distributed Systems Group, TU Wien, Vienna, Austria
{d.nguyen,truong,e.copil,d.le,d.moldovan,dustdar}@dsg.tuwien.ac.at

Abstract. The Data-as-a-Service (DaaS) model enables data analytics providers to provision and offer data assets to their consumers. To achieve quality of results for the data assets, we need to enable DaaS elasticity by trading off quality and cost of resource usage. However, most of the current work on DaaS is focused on infrastructure elasticity, such as scaling in/out data nodes and virtual machines based on performance and usage, without considering the data assets' quality of results. In this paper, we introduce an elastic data asset model for provisioning data enriched with quality of results. Based on this model, we present techniques to generate and operate data elasticity management process that is used to monitor, evaluate and enforce expected quality of results. We develop a runtime system to guarantee the quality of resulting data assets provisioned on-demand. We present several experiments to demonstrate the usefulness of our proposed techniques.

1 Introduction

To provide flexible access to vast amounts of data, several types of Data-as-a-Service (DaaS) have emerged to allow users to execute data analytics atop a vast, rich set of data sources, such as Azure's Data Market¹, Infochimps², Factual³, and a number of research systems [1–3]. Many of these systems support the concept of elasticity by scaling data nodes, re-assigning data partitions and re-configuring data clusters to automatically adapt to dynamic changes in their workload [1–3]. Overall, these systems support the elasticity of data services at the infrastructure level. The question of how to ensure the quality of resulting analytics, covering quality of data, prices, analytics time, and forms of outputs for data analytics, has not been in the focus of existing elastic data systems.

To solve this problem, we examine another aspect of elasticity in data analytics within DaaS by considering how we could provide data assets resulting from data analytics in an elastic manner. That is, data consumers must be able to request, obtain and utilize data assets with different quality and cost based

¹ <http://datamarket.azure.com>.

² <http://www.infochimps.com>.

³ <http://factual.com>.

on their requirements. For example, if the quality of data associated with the resulting data assets is high, the cost for the resulting data assets might be increased. To achieve this in an elastic manner, besides dealing with the elasticity of computing resources for data processing, the DaaS provider must have mechanisms to deal with the elasticity of quality of data and cost.

In our approach, a data provider can provision a data asset based on quality of results (QoRs) by utilizing DaaS. Within DaaS there are different data analytics functions, which can be represented as a workflow of data analytics tasks, produces a data asset. A QoR model [4] can be used to specify, e.g., the quality of the data asset, the output form of the data asset, the time to deliver the data asset, and the price of the data asset. To ensure the QoR of the data asset, the DaaS provider needs to deploy, control, monitor not only its underlying computing systems (i.e., virtual machine), but also the data used to offer data assets. We refer to process having the above-mentioned features as data elasticity management process (DEP).

In this paper, we introduce a novel model of elastic data asset to associate the data asset with its QoR. Elastic data asset can have states, w.r.t. quality and cost, that can be changed over time to reflect the QoR in the interest of the data consumer. Based on that, we present techniques to support generating DEP to ensure QoR for data asset and the runtime environment to operate DEP for DaaS. In this paper, we illustrate our approach through real-world applications of data assets provisioning near-real time data.

The rest of this paper is organized as follows: Sect. 2 presents the motivation for our work. Section 3 introduces the model for elastic data assets. Section 4 presents techniques to generate data elasticity management process and its runtime. Section 5 presents experiments. Section 6 discusses related work. Finally, we conclude the paper and discuss our future work in Sect. 7.

2 Motivation and Approach

Let us consider a scenario with a GPS DaaS provider and several DaaS consumers, such as public transportation and taxi companies. They want to sell and buy near-real time GPS data of vehicles in the HoChiMinh City. The provider owns this data source and wants to sell potential data assets to DaaS consumers, trading off (i) accuracy of moving-vehicle – abbreviated by *vehicleAcr* – the percentage of on-street vehicles over the total number of vehicles, (ii) accuracy of speed of vehicles – *speedAcr* – the percentage of vehicles have the difference of measured speed and estimated speed higher than a threshold, *deliveryTime* – the minimum time to deliver data asset in seconds, and cost. The DaaS provider wants to sell data assets in a flexible way based on a QoR associated with the data assets. For example, a data asset is sold with a $QoR = \{vehicleAcr \geq 81\%, speedAcr \geq 81\%, deliveryTime \leq 55(seconds)\}$, and cost per a window of data asset €0.09}. Furthermore, a customer might accept lowered values of *vehicleAcr* and *speedAcr* and a higher *deliveryTime*, if the price is reduced.

Currently, it is challenging to build such a DaaS that can fulfill the above requirements from both the provider and customer perspectives. First, the provider

must provision data assets, which have business values, from suitable data analytics processes (e.g., enriching near-real time GPS data with estimated average speeds of vehicles in different areas). Second, the DaaS provider must define metrics w.r.t QoR for the data asset and for each type of data asset, the provider can choose many metrics associated with it. Third, the provider must develop DEP for monitoring and controlling quality of data, performance of DaaS and resource usage to ensure QoR. Finally, the provider must develop a suitable runtime environment for executing the DEP together with data analytics process.

One major issue that arises from the above scenario is that the DEP are tightly coupled to the metrics in specified and expected QoR, which makes the DEP very difficult to be extended with other QoR. Moreover, information of data analytics, which is used to provision data asset, must be considered when creating DEP. It is also difficult for the DaaS provider to modify the DEP, after the DaaS has already been deployed. We need methods to create DEP that can be reusable and extensible w.r.t. changes of the QoR metrics associated with data asset. However, current models of data services [1–3,5] are not yet associated with QoR that can support the above-mentioned requirements. These challenges motivate us to develop a technique to support generating DEP. The goal is to support the DaaS provider easily provisioning data asset from data analytics and QoR. Based on these inputs, we can support to generate DEP to achieve the elasticity of the provisioned data assets.

3 Elasticity Model for Data Assets

3.1 Data Assets and Their Quality of Results

A provider might utilize different data sources to offer data assets. We assume that a data item can be a record in a relational database, a document in document-based database or a key-value pair in key-valued database. An analytics of a set of data items will produce a data asset. In our work, we support to provision data assets from batch and near real time data analytics; a near real time data asset is delivered to customers with a predefined time window. A window of data asset also includes a set of data items. Let DAF be a set of data analytics functions defined based on a data model for data sources. Such function can be simple, e.g., including several data queries, or complex, e.g., including different data analysis algorithms. For provisioning data assets from the data sources, each data asset can be produced by a data analytics function (DAF) $dataAsset = \{daf, daf \in DAF\}$. The DaaS provider can provision data assets they want to sell by defining and executing the DAF. For example, a DAF may include activities for (i) reading streaming GPS data, (ii) clustering vehicle location, (iii) estimating the average speed of vehicles in each cluster, and (iv) outputting data in the form of comma-separated values (csv).

Data assets offered to data consumers can be characterized by many metrics. These metrics are rich and dependent on data sources and DAF, such as data accuracy, data completeness and data consistency [6]. Moreover, the monetary value of the data assets may depend on the values of these quality metrics.

To support the DaaS provider to present the expected values of these metrics and cost, we reuse the QoR model in [4, 7]. The QoR model includes a set of metrics, a set of QoR elements (qElement) and a form of data asset.

- A set of metrics measures quality of data assets. Each metric can be accessed/adjusted by primitive actions detailed in the next subsection.
- A qElement is defined as a set of conditions established based on these metrics in specific ranges and a specific cost for data assets.
- The form of data assets indicates the format of the output asset (e.g., comma-separated values or a bar chart) resulted from DAF.

We choose this QoR model to associate QoR metrics⁴ to data asset because this model is easy for DaaS provider to present the expectation of quality and cost of data asset together with DAF.

3.2 Data Elasticity Management Process

Figure 1 presents the relationship between data assets, DAFs and DEP. By executing the **data analytics function 1**, we have *data asset y*. The provider would like to sell *data asset y* to their customers. To ensure the QoR of this data asset, a data elasticity management process is invoked. A data elasticity management process includes sub-processes to monitor/control quality of data and performance (e.g., *deliveryTime*), and a resource control plan to manage resource usage at infrastructure level when executing these sub-processes.

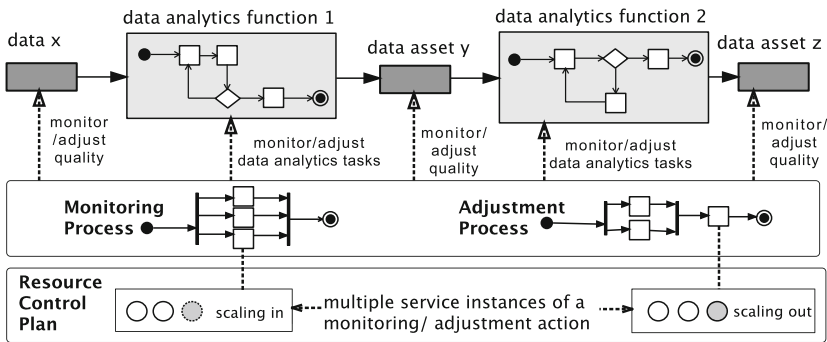


Fig. 1. Ensuring QoR for data assets by using data elasticity management process

There are different approaches to apply DEP: (i) improve the quality of *data x* which is used for determining *data asset y* from **data analytics function 1**, (ii) adjust *data analytics function 1* by tuning parameters, plugging sub-processes or replacing process fragments to improve the quality of *data asset y* better, and/or (iii) improve the *data asset y* to produce a better data asset by separate processes. In this paper, we support the third method, through which we can

⁴ <https://github.com/tuwiendsg/EPICS/blob/master/depic/examples/qor/qor.yml>.

ensure QoR of data asset produced by DAF. The general principle is that we store result data assets from DAF into a data buffer, then we use the sub-processes to ensure quality of data before delivering the data assets to customer. The sub-processes include many actions organized on workflows. We refer to these actions as primitive actions. A primitive action can be used to perform data assessment and quality adjustment, for example, applying regression algorithms [6] to smooth the values of vehicle speeds to increase data accuracy. QoR of data assets also depends on underlying computing infrastructure resources, e.g., delivery time and throughput of data asset. To adjust the values of these metrics, we can carry out primitive actions such as dynamic provisioning infrastructure resources to distribute workloads for many processing services [1]. For example, scaling out computational resources for data cleansing services can help to increase the throughput.

3.3 Managing Primitive Actions

Figure 2 shows the model for capturing primitive action metadata⁵. Primitive actions to monitor quality of data have to return the values of quality of data assets and can be executed in parallel, while primitive actions to adjust the quality of data do not return values and have to be executed in a particular order to avoid data corruption by overwriting and achieve semantic of adjustment processes. Therefore, a primitive action can be an adjustment action (i.e., an action to adjust the quality of data asset), a monitoring action (i.e., an action to assess quality of data asset) or a resource control action (i.e., an action to scale in/out monitoring/adjustment services at runtime). At runtime, a primitive action is executed through an invocation of an adjustment/monitoring service. A primitive action call will invoke its corresponding service. To deal with different situations of quality of a data asset, an adjustment action can have multiple adjustment cases. Each adjustment case is specified by the parameters of its adjustment action. To determine the right adjustment case, we use information

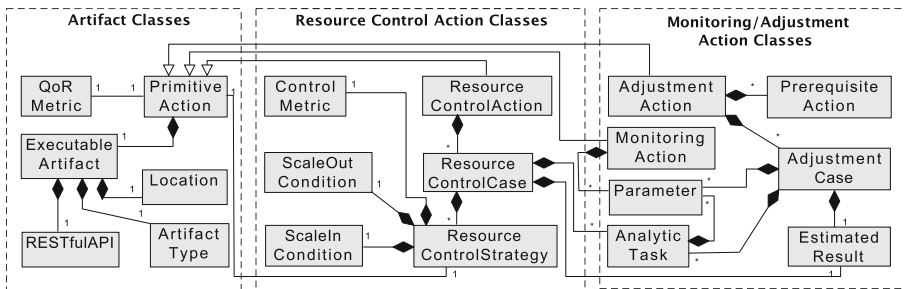


Fig. 2. Model of primitive action metadata

⁵ An example of primitive action metadata can be found here <https://github.com/tuwinds/EPICS/blob/master/depic/examples/pam/pam.yml>.

about expected QoR in PAM and analytic tasks in DAF. For example, an adjustment action, which removes missing-value records in a data asset to improve data completeness, has an estimated result 100%. For another example, an analytic task in DAF employs k-means with the stop condition after 5 loops that may lead to a bad data asset because the convergence does not approach global optimum [8]. Previous studies proposed methods to determine metadata for adjustment cases, for example, selecting input parameters for decision trees [9], or selecting input parameters for clustering data [8]. These studies can help to indicate right parameters from estimated QoR metrics and analytics tasks for adjustment cases. We assume that data about primitive action metadata results from these studies, which are out of the scope of our study. A prerequisite action is used to decide which adjustment actions are executed before another one. A resource control strategy has conditions for metrics at system level (e.g., cpuUsage) to scale in/out monitoring/adjustment services.

3.4 Elastic Data Asset

Considering a data asset (da) resulted from a DAF, we use a monitoring process to determine if the da is delivered with the expected qElement or not. If not, we apply an adjustment process to this da to create another da which will meet the expected qElement. To model the changes of QoR of the da by predetermined monitoring and adjustment processes, we need to model elasticity states (i.e., states w.r.t qElement within the QoR) associated with the da and DEP.

Equation 1 defines an elastic state, which is a binary vector of conditions associated with each metric being monitored. For metric i , let em_i be the current evaluated metric value, while c_{j_i} gives condition j_i for metric i . This eState is evaluated to true if the conjunction over the $eState[j_i]$, considering the current metric values em_i , is evaluated to true. The set of all eStates associated with a data asset, ES_{da} , is given by the eStates obtained from the combination of all conditions available for all metrics specified in the QoR, as defined in Eq. 2.

$$eState_{da}[j_1 \dots j_n] = \{[c_{j_1}(em_1), \dots, c_{j_n}(em_n)] \mid em_i \in EM_{qor}, c_{j_i} \in CD_{qor}\} \quad (1)$$

$$ES_{da} = \{eState_{da}[j_1 \dots j_n] \mid \forall [j_1 \dots j_n] \in combinations(j_x), j_x \in NumberOfConditions(Metric_x, qor)\} \quad (2)$$

The eda , defined in Eq. 3, is composed of da , which is obtained by applying a daf , the ES_{da} defined in Eq. 2, the set of data elasticity management processes DEP , and the initial and final eStates, $eState_{in}$ and $eState_{fi}$. The eStates are managed by a set of data elasticity management processes DEP .

$$eda = (da, ES, eState_{in} \in ES, DEP, eState_{fi} \in ES) \quad (3)$$

The data elasticity management processes (*DEP*) associated with an *eda* includes a monitoring process (p_m), adjustment processes (p_c) and resource control plan (p_r). A p_m is used to determine $eState_{in}$ of da, while p_c and p_r are used to do transition of the da from an $eState_{in}$ to an $eState_{fi}$ in *ES*:

- Monitoring process: let MA be the set of monitoring actions. Each monitoring action is mapped to a specific monitoring service to measure value of a specific QoR metric. Thus, a monitoring process of a da is defined as $p_m = \{ma(parameters)\}, ma \in MA$; $eState = p_m(da)$; and $parameters$ denotes parameters for the monitoring actions.
- Adjustment process: let CA be the set of adjustment actions. Each adjustment action is mapped to a specific adjustment service to adjust quality of a specific QoR metric. An adjustment process can be defined as $p_c = (\{ca(parameters)\})$, where $ca \in CA$.
- Resource control plan: let RA be the set of resource control strategy. Each resource control strategy is used to control a metric at infrastructures to ensure the quality of a specific QoR metric. A resource control plan can be defined as $p_r = (\{ra\})$, where $ra \in RA$.

4 Generating and Operating Data Elasticity Management Processes

4.1 Generating Data Elasticity Management Processes

Based on the model of *eda*, our approach to generate DEP for data asset includes 3 steps: (i) generating a monitoring process to understand the quality of data assets, (ii) generating an adjustment process to adjust the quality of data assets, and (iii) generating a resource control plan during the execution of the processes.

Algorithm 1. Algorithm to generate data elasticity management processes

```

1: function GENERATE DATA ELASTICITY MANAGEMENT PROCESSES(qor, daf, pam)
2:   listOfMonitoringAction =findMonitoringActions(qor.metricList(), pam)
3:   monitoringProcess =parallelizeMonitoringActions(listOfMonitoringAction)
4:   finalEStateSet =decompose(qor.qElements(), pam)
5:   for each eState in finalEStateSet do
6:     adjustmentCases =findAdjustmentCases(eState, pam, daf)
7:     adjustmentProcess =buildWorkflow(adjustmentCases, pam)
8:     resourceControlPlan=findControlStrategies(eState, pam, daf)
9:     elasticProcesses =
10: new ElasticProcesses(eState, monitoringProcess, adjustmentProcess, resourceControlPlan)
11:   end for
12: end function

```

Algorithm 1 describes the algorithm to generate DEP. The inputs for this algorithm include a QoR, a DAF and the primitive action metadata (PAM). Based on the list of metrics in the QoR and the monitoring actions associated with these metrics in PAM, the algorithm generates a monitoring process by

organizing the monitoring actions in parallel (line 2–3). The final *eState* set of a data asset is determined through decomposing ranges of values of QoR metrics of the *qElements* into conditions of estimated results of corresponding metrics (line 4). For each *eState* in the final *eState* set, the algorithm finds primitive actions based on its corresponding QoR metrics of conditions in the *eState*. If the primitive action is a type of adjustment action, the algorithm finds adjustment cases by matching (i) the conditions in the *eState* with estimated results in PAM and (ii) analytic tasks in the DAF with the ones in PAM (line 6). The adjustment cases are found if both (i) and (ii) are matched. An adjustment process is built from the adjustment action list and prerequisite actions in PAM (line 7). The algorithm creates sub-workflows of the adjustment actions by connecting the prerequisite actions in sequence. Then, these sub-workflows are connected by parallel gateways. If the primitive action is a type of resource control action, the algorithm finds resource control strategies by matching (iii) the conditions in the *eState* with estimated results in PAM and (iv) analytic tasks in the DAF with the ones in PAM (line 8). The resource control strategies are found and added to the resource control plan if both (iii) and (iv) are matched. The resource control plan is used to create SYBL control strategies [10] for scaling in/out computing resources for data elasticity operations.

4.2 Runtime for Data Elasticity Management Process

To support the generation and operation of DEP, we have implemented the *Tooling* and the *Runtime*, as shown in Fig. 3. *Tooling* allows the DaaS provider to define DAF and QoR used for generating DEP and to tune parameters’ values of monitoring/control actions in DEP. The *Cloud Service Specification Generator* utilizes COMOT services [11] API to determine the non-existing monitoring and adjustment services in the cloud infrastructure and generate a cloud service specification describing the deployment of DEP and other runtime services used for QoR enforcement, such as *Monitoring/Adjustment Services*, *Data Asset Loader* and *EDA Repository*, and resource control plan. In our runtime, the *Data Asset Loader* is responsible for getting data assets from data analytics functions and storing them into the *EDA Repository* from which data assets will be passed

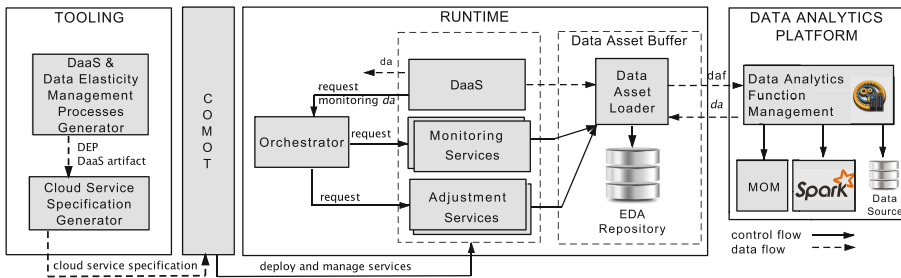


Fig. 3. The architecture of the runtime of data elasticity management processes

to enrichment actions. The *Orchestrator* executes monitoring process, handling validation and applying appropriate adjustment process. In our prototype⁶, we interface to different *Data Analytics Platforms*. These platforms are responsible for processing DAF and returning unqualified data assets.

5 Evaluation

5.1 Experiment Settings

We use the scenario described in Sect. 2 with near-real time GPS data of vehicles in the HoChiMinh City. We obtained this 1.17 GB real data from our research collaborators and emulated real-time data sources by sending historical GPS data to scalable message oriented middleware (MOM) located in the same cloud infrastructure with our runtime services. Figure 4 shows two experimental DAFs used to provision data assets. *daf1* gets near-real time GPS data from MOM, clustering locations of vehicles based their latitude and longitude - window size of 5000 data items, and estimating vehicle speed in each cluster. *daf2* is an extension of the *daf1* that checks if the current estimate speed is over a threshold, historical GPS data will be used to estimate the average vehicle speed, enrich data with address and output data. The DaaS consumers might be interested in using the data asset 1 to detect potential traffic congestion and the *DataAsset2* to find causes of traffic congestion.

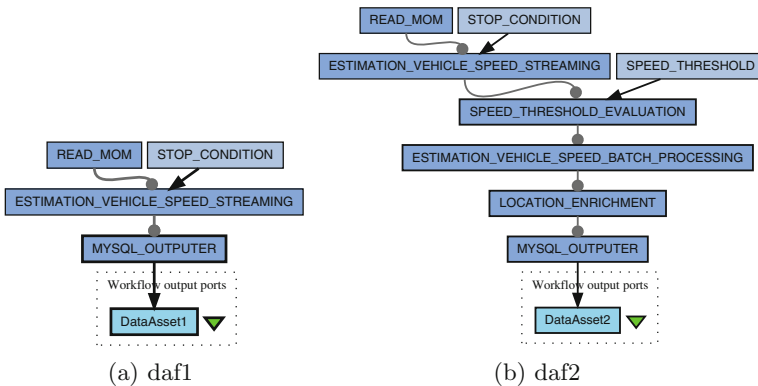


Fig. 4. Data Analytics functions for provisioning GPS data

We have two phases in provisioning data assets. First, we deployed DAFs which continuously deliver initial data assets to the Data Asset Buffer; the execution of DAFs represents a streaming data analytics system. Second, when a consumer requests a data asset through a DaaS, the corresponding generated DEP will take streaming initial data assets in the runtime *Data Asset Buffer* and then apply monitoring and quality enrichment before returning suitable

⁶ Available at: <https://github.com/tuwiendsg/EPICS/tree/master/depic>.

data assets to the consumer. In the first phase, all customers should share the operation cost, whereas in the second case, each customer pay only the cost due to the delivery of its data assets.

The QoR of the data assets are measured by the following metrics: *speedAcr* – accuracy of speed of vehicles, i.e., the percentage of vehicles have the squared deviations of speed higher than a threshold, *vehicleAcr* – accuracy of moving-vehicle, i.e., the percentage of on-street vehicles over the total number of vehicles, *throughput* – the average number of data assets delivered per second, abbreviated as das/s, and *deliveryTime* – the minimum time (in seconds) to deliver data asset in the second phase.

For primitive action metadata (PAM), we assume that domain experts use the model $primitiveAction = \{goRMetric, estimatedResult, analyticTask\}$ to fill in right primitive action for specific QoR metric, estimated result and analytic task. Depending on different estimated results and analytic tasks, different primitive actions and their parameters are used.

5.2 Generating Data Elasticity Management Processes

We use different QoRs, DAFs and primitive action metadata to evaluate our proposed generation technique, as shown in Table 1. We evaluate the generated DEP using completeness. The completeness has value yes in case the DEP have complete monitoring process, adjustment processes and resource control plans to ensure QoR of the data asset, and value no in case the DEP are unable to ensure QoR of the data asset because at least one of monitoring process, adjustment processes and resource control plans has missing or conflict data.

Table 2 summarizes the results of generating DEP in 5 cases:

- case 1: Because the conditions in the $qElement$ set match the estimated results in the primitive action metadata *pam1* completely, and *pam1* has complete information of primitive actions, the generated DEP are complete.
- case 2: We use the $qElement2$, which has a condition of *vehicleAcr* is $[91,95]$. This condition is a subset of *estimatedResult* $[81,100]$. Therefore, the estimated result in ranges $[81,90]$ and $[96,100]$ cannot be satisfied and the generated adjustment process is not complete.
- case 3: We use *daf2*, which has stopCondition of K-means is different from the ones in the adjustment cases, leading to the incompleteness of the generated resource control plan.
- case 4: *pam2* has an adjustment action for *vehicleAcr*, however, the estimated result is missing. So that the adjustment processes are not complete.
- case 5: The resource control plans of *throughput* and *deliveryTime* are conflicted because their resource control strategies control the same monitoring/adjustment services at runtime. Therefore, we have to choose another resource control plan to deal with these metrics manually.

We can see that our techniques could generate complete DEP when information of primitive actions are complete. In case of missing information of primitive actions or conflicts in DEP, the DEP can be customized manually. In future, we will develop an algorithm to automatically resolve these conflicts.

Table 1. Summary of QoR, primitive action metadata and DAF

	#	qElement	[speedAcr(%)];[vehicleAcr(%)];[deliveryTime(s)]; [throughput(das/s)];cost(€)
QoR	qor1	qElement1	[81,100];[81,100];[0,55];[]:[0.007]
		qElement2	[81,100];[81,100];[56,∞];[]:[0.006]
		qElement3	[61,80];[61,80];[0,55];[]:[0.006]
		qElement4	[41,60];[41,60];[0,55];[]:[0.005]
		qElement5	[21,40];[21,40];[0,55];[]:[0.004]
	qor2	qElement1	[81,100];[91,95];[0,55];[]:[0.007]
		qElement2	[81,100];[81,100];[56,∞];[]:[0.006]
		qElement3	[61,80];[61,80];[0,55];[]:[0.006]
		qElement4	[41,60];[41,60];[0,55];[]:[0.005]
		qElement5	[21,40];[21,40];[0,55];[]:[0.004]
	qor3	qElement1	[81,100];[81,100];[0,55];[1.05,∞];[0.007]
		qElement2	[81,100];[81,100];[56,∞];[0,1.04];[0.006]
		qElement3	[61,80];[61,80];[0,55];[1.05,∞];[0.006]
		qElement4	[41,60];[41,60];[0,55];[1.05,∞];[0.005]
		qElement5	[21,40];[21,40];[0,55];[1.05,∞];[0.004]
PAM	#	primitive action	[associatedQoRMetrics];[estimatedResult]; [analyticsTask - parameter:value]
	pam1	adjustmentAction1	[speedAcr];[81,100];[kmeans - stopCondition:5]
		adjustmentAction2	[vehicleAcr];[81,100];[]
		resourceControlAction1	[deliveryTime];[0,55];[]
		resourceControlAction2	[throughput];[1.05,∞];[]
	pam2	adjustmentAction1	[speedAcr];[81,100];[kmeans - stopCondition:5]
		adjustmentAction2	[vehicleAcr];[];[]
		resourceControlAction1	[deliveryTime];[0,55];[]
		resourceControlAction2	[throughput];[1.05,∞];[]
DAF	#	DAF task	[analyticsTask - parameter:value]
	daf1	estimation vehicle speed	[kmeans - stopCondition:5]
	daf2	estimation vehicle speed	[kmeans - stopCondition:10]

Table 2. Completeness of DEP in different cases

	case 1	case 2	case 3	case 4	case 5
QoR	qor1	qor2	qor1	qor1	qor3
Primitive action metadata	pam1	pam1	pam1	pam2	pam1
Data analytics function	daf1	daf1	daf2	daf1	daf1
Completeness of the processes	Yes	No	No	No	No

5.3 Operating Data Elasticity Management Processes

We evaluate the generated DEP in **case 1** at runtime. In the evaluation, we show 3 aspects of elasticity including resource (i.e., virtual machine), quality

(i.e., *vehicleAcr*, *speedAcr* and *deliveryTime*) and cost (i.e., processing cost and data asset cost). We use 1 VM (3GB RAM, 2 vCPUs, 40GB Disk) for *COMOT* services. We use 1 *m1.small* VM (1GB RAM, 1 vCPU, 40GB) for *MOM*. We use 1 VM (7GB RAM, 4 vCPUs, 40GB Disk) for *Tooling*, *Orchestrator*, *Data Asset Loader* and *Data Analytics Function Management*. We use 4 VMs for monitoring/adjustment services at the beginning. Each monitoring/adjustment service runs on 1 *m1.small* VM. We test the execution of the generated DEP with 5 concurrent DaaS consumers in case of using $\text{consumerRequirement1} = \{\text{vehicleAcr} \geq 81\%, \text{speedAcr} \geq 81\%, \text{and } \text{deliveryTime} \leq 55(s)\}$ and $\text{consumerRequirement2} = \{\text{vehicleAcr} \geq 61\%, \text{speedAcr} \geq 61\%, \text{and } \text{deliveryTime} \leq 55(s)\}$. To study cost elasticity, we defined the data asset cost as follows:

$$\text{cost}_{da} = \sum_{i=1}^{nbMetrics} \text{unitCost}(qorMetric_i) * w_i * \left(\sum_{j=1}^{nbCond_i} (j * qElement_{cond_j}) \right) \quad (4)$$

$qElement_{cond_j}$ is a boolean condition associated with $qorMetric_i$, and w_i are weighted factors. For each $qorMetric_i$ in a $qElement$, only one $qElement_{cond_j}$ has value 1 and the others have value 0. The data asset cost depends on the values of *vehicleAcr* and *speedAcr*, which are divided into 5 ranges to present quality of data from low to high. This cost also depends on the processing time, including *analyticTime* in the first phase and *deliveryTime* in the second phase. From this general function, we have 5 functions for data asset cost assumptions - f1 and f2 ($w_{processingTime} = 1, w_{vehicleAcr} = 0, w_{speedAcr} = 0$), f3 and f4 ($w_{processingTime} = 0, w_{vehicleAcr} = 0, w_{speedAcr} = 1$) and f5 ($w_{processingTime} = 0.5, w_{vehicleAcr} = 0.25, w_{speedAcr} = 0.25$). We use the unit cost for *speedAcr* and *vehicleAcr* as €0.0002, the unit cost for machines in the data analytics phase as 0.104 EUR (the same as the cost of t2.large instance⁷), and the unit cost for machines in the enrichment phase as €0.026 (the same as the cost of t2.small instance).

Figure 5 shows *deliveryTime* of data asset with different consumer requirements. *deliveryTime* is controlled under the range [0,55] from the 46th and 14th window of data in case of using *consumerRequirement1* and *consumerRequirement2*, respectively. Figure 6(a) shows the values of *speedAcr* in cases of using and not using the DEP: the values of *speedAcr* are always in ranges [81,100] in case of using the DEP but not in the case of not using the DEP.

Figures 5 and 6(a) show data asset cost defined by f1, f2, f3 and f4. We see the relationships between the values *deliveryTime*/*speedAcr* and data asset cost. When the values of QoR do not meet expected values, the data asset cost needs to be lowered. Figure 6(b) shows the relationship between processing cost and data asset cost from f5 in cases of using *customerRequirement1* and *customerRequirement2*. The processing cost for an data asset is the sum of the quality adjustment cost and data analytic cost. The quality adjustment cost is calculated by multiplication of *deliveryTime*, the number of used VMs and the unit cost of t2.small instance. The total data analytic cost is calculated by the multiplication

⁷ <http://aws.amazon.com/ec2/pricing/>.

of analytic time, the number of used VMs and a unit cost of t2.large instance so we assume the data analytic cost per data asset equals to the total data analytic cost divided by the number of consumers. We see that the processing cost at the beginning is high because *deliveryTime* is too long. Then, the processing cost increases (i.e., from the 3th window of data asset to the 35th window of data asset) because the adjustment/monitoring services scale out continuously (i.e., more VMs are used). After that, the cost of VMs remains unchanged because *deliveryTime* is stable and there is no need of scaling in/out actions. Figure 6(b) shows that, with f5 for consumer requirement 2, compared with consumer requirement 1, the processing cost has a higher degree of fluctuations than that of data asset cost, suggesting in certain situations the provider spends more operational costs. We cannot conclude which is the best function for data asset cost, but support the provider to decide an appropriate cost functions.

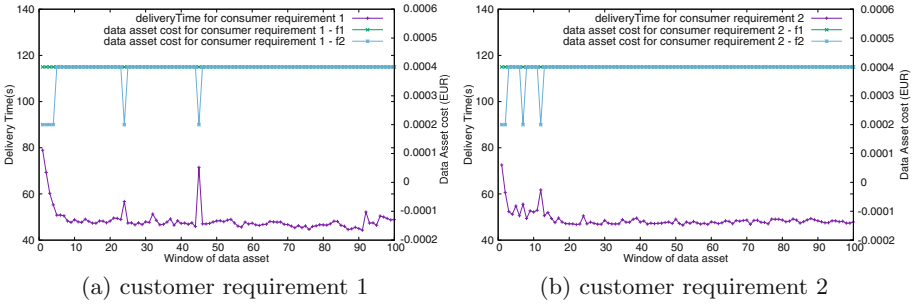


Fig. 5. deliveryTime and data asset cost

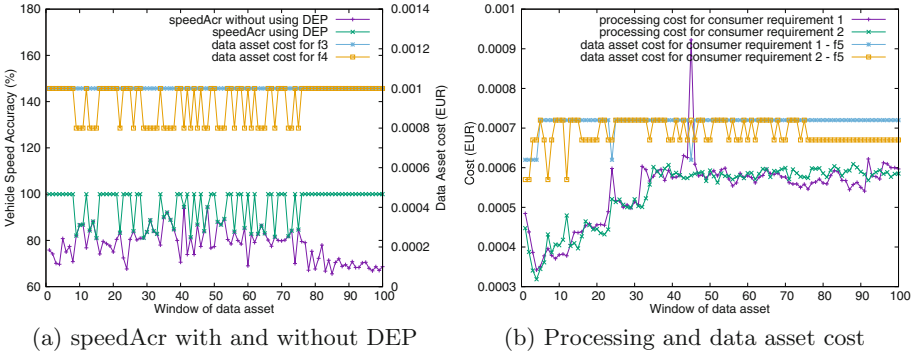


Fig. 6. speedAcr, processing cost and data asset cost

6 Related Work

Elasticity Model of Data Provision: Existing studies have been introduced to support certain elasticity models for data provisioning. Examples are ElasTras which

supports dynamic partition reassignment [1], functional requirement changes using a database schema evolution rule [2], reconfiguration of data cluster responding to workload changes [5], and changes of mapping of identifiers to storage processes under heavy load [3]. In general, these works support elastic data distribution by dynamically adding/removing data nodes and moving data partitions between data nodes. Our research objective is different in the sense that our data model is introduced to abstract data asset and DEP to ensure QoR.

Data Elasticity Management Processes for Quality of Analytics: Although there exist techniques to enable DEP [7,12,13], they do not support runtime QoR-aware delivery and runtime environment for DEP. Our previous Drain supports selecting and configuring process fragments to ensure QoR of data [7]. However, it does not consider factors which impacts on the QoR of data from analytics functions and does not address resource controls at runtime; it also does not generate DEP for DaaS. Hauder [14] presented a framework to generate data processes from an abstract process, but this approach uses only one criterion (valid/invalid) to evaluate generated process and do not consider QoR of data and elasticity at runtime. Liu et al. presented the requirement for data service composition as nested tables [12] and proposed an algorithm for data service composition based on links and weights of links between actions. Wang et al. proposed a model to specify a pre-defined data process and requirements of quality of services as well as cost and presented an improved version of a genetic algorithm to select data-intensive service when generating service composition [13]. This algorithm optimizes the service selection process based quality of services and cost (e.g., data cost, access cost, and transfer cost). Different from existing approaches, we use DAF to provision a data asset and a QoR model to present requirements of this data asset. We develop a technique to generate DEP to monitor, adjust quality of data asset and control resource at runtime. Moreover, previous studies have not investigated elasticity when generating service composition, while we consider elasticity at runtime.

7 Conclusions and Future Work

Data assets produced from data analytics functions should be provided based on different quality of results in an elastic manner. In this paper, we show that, with appropriate knowledge about primitive actions for monitoring, adjustment and resource control, we can support the provider to generate DEP as well as can leverage underlying elastic platforms to manage and operate DEP to provision QoR-aware data assets. We also support the provider to study data asset cost functions based on QoR and cost of resource usage.

We are currently testing several experiments of generated DEP with many other types of data. We are working on optimizing the execution of data analytics functions. Moreover, we will develop a programming framework to support the DaaS provider to easily build services for primitive actions.

Acknowledgment. This work is supported by the European Commission in terms of the CELAR FP7 project (FP7-ICT-2011-8 #317790).

References

1. Das, S., Agrawal, D., El Abbadi, A.: Elastras: an elastic, scalable, and self-managing transactional database for the cloud. *ACM Trans. Database Syst.* **38**, 5:1–5:45 (2013)
2. Ishida, Y.: Scalable variability management for enterprise applications with data model driven development. In: *International Software Product Line Conference Co-located Workshops, SPLC 2013*, pp. 90–93. ACM, New York (2013)
3. Unterbrunner, P., Alonso, G., Kossmann, D.: High availability, elasticity, and strong consistency for massively parallel scans over relational data. *VLDB J.* **23**(4), 627–652 (2014)
4. Truong, H.L., Dustdar, S.: Principles of software-defined elastic systems for big data analytics. In: *International Conference on Cloud Engineering, IC2E 2014*, pp. 562–567. IEEE Computer Society, Washington (2014)
5. Cruz, F., Maia, F., Matos, M., Oliveira, R., Paulo, J.A., Pereira, J., Vilaça, R.: Met: workload aware elasticity for nosql. In: *European Conference on Computer Systems, EuroSys 2013*, pp. 183–196. ACM, New York (2013)
6. Batini, C., Cappiello, C., Francalanci, C., Maurino, A.: Methodologies for data quality assessment and improvement. *ACM Comput. Surv.* **41**, 16:1–16:52 (2009)
7. Murguzur, A., Schleicher, J.M., Truong, H.-L., Trujillo, S., Dustdar, S.: DRain: an engine for quality-of-result driven process-based data analytics. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) *BPM 2014*. LNCS, vol. 8659, pp. 349–356. Springer, Heidelberg (2014)
8. Guo, J.J., Luh, P.: Selecting input factors for clusters of gaussian radial basis function networks to improve market clearing price prediction. *IEEE Trans. Power Syst.* **18**, 665–672 (2003)
9. D’heygere, T., Goethals, P.L., Pauw, N.D.: Use of genetic algorithms to select input variables in decision tree models for the prediction of benthic macroinvertebrates. *Ecol. Model.* **160**, 291–300 (2003). Modelling the structure of aquatic communities: concepts, methods and problems
10. Copil, G., Moldovan, D., Truong, H.L., Dustdar, S.: Sybl: an extensible language for controlling elasticity in cloud applications. In: *13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 112–119. IEEE (2013)
11. Truong, H.L., Dustdar, S., Copil, G., Gambi, A., Hummer, W., Le, D.H., Moldovan, D.: Comot-a platform-as-a-service for elasticity in the cloud. In: *IEEE International Workshop on the Future of PaaS, colocated with IC2E*. IEEE (2014)
12. Liu, C., Wang, J., Han, Y.: Situation-aware data service composition based on service hyperlinks. In: Huang, Z., Liu, C., He, J., Huang, G. (eds.) *WISE Workshops 2013*. LNCS, vol. 8182, pp. 153–167. Springer, Heidelberg (2014)
13. Wang, L., Shen, J., Luo, J., Dong, F.: An improved genetic algorithm for cost-effective data-intensive service composition. In: *2013 Ninth International Conference on Semantics, Knowledge and Grids (SKG)*, pp. 105–112 (2013)
14. Hauder, M., Gil, Y., Liu, Y.: A framework for efficient data analytics through automatic configuration and customization of scientific workflows. In: *IEEE 7th International Conference on E-Science, e-Science 2011*, 5–8 December 2011, Stockholm, Sweden, pp. 379–386 (2011)

Cloud Services Management

Supporting Cloud Service Operation Management for Elasticity

Georgiana Copil^(✉), Hong-Linh Truong, and Schahram Dustdar

Distributed Systems Group, Vienna University of Technology, Vienna, Austria
{e.copil,truong,dustdar}@dsg.tuwien.ac.at

Abstract. Complex cloud services rely on various IaaS, PaaS, and SaaS cloud offerings. Fast (re)deployment and testing cycles, and the rapidity of changes of various dependent infrastructures and services, imply a need for continuous adaptation. Although software-based elasticity control solutions can automate various decisions through intelligent decision-making processes, in many cases, such adaptation requires interactions among different cloud service provider employees and among different providers. However, decisions from stakeholders and elasticity software controllers should be seamlessly integrated.

In this paper, we analyze the needs of service providers and the possible interactions in elasticity operations management that should be supported. We focus on interactions between service provider employees and elasticity controllers, and propose novel interaction protocols considering various organization roles and their concerns from the elasticity control point of view. We introduce the elasticity Operations Management Platform (eOMP) which supports seamless interactions among service provider employees and software controllers. eOMP provides elasticity directives to enable notifications for complex elasticity issues to be solved by service provider employees, and the necessary mechanisms for managing cloud service elasticity. Our experiments show that service provider employees can easily interact with elasticity controllers, and, according to their responsibilities, take part in the elasticity control to address issues which may arise at runtime for complex software services.

1 Introduction

A service deployed in the cloud can make use of various resources and services offered by cloud providers, and can be very dynamic at run-time. The cloud is one of the most dynamic environments: providers can change their cost schemas, and the offered service characteristics, from one day to another. Although in this environment automated controllers are necessary, given this high dynamism, it might be necessary for service stakeholders to re-examine the desired behavior of a cloud service, and their interactions with other stakeholders (e.g., cloud providers, or data providers). For instance, whenever the load of the cloud service dramatically changes, the normal “safety requirements” (e.g., do not exceed

This work was supported by the European Commission in terms of the CELAR FP7 project (FP7-ICT-2011-8 #317790).

a specific cost value) might not hold. For these situations, service providers have employed analysts to oversee the control process, and detect when such a case is encountered by a controller. A much better solution would be that the controller itself notifies the responsible person with the encountered situation (e.g., unexpected behavior or service health issues). In such situations, the employees responsible for the detected cases, once notified, should be able to easily interact with the controller to solve the detected issues. This type of “human-in-the-loop” based control not only improves the runtime elasticity customization capabilities, but also empowers service providers with more control over their services and automated elasticity controllers. Thus, for obtaining service elasticity at runtime, the service provider needs this kind of support in its operation phase (i.e., from service management lifecycle), in order to carry out the service operation processes in the cloud environment. Although currently several solutions provide elasticity control of cloud services (e.g., [1, 2]), service provider employees are not included in the elasticity control process.

For addressing the challenges above, in this paper we propose adding *roles* (i.e., service provider employees) as first class entities in cloud service elasticity control loops. Based on the roles, we define necessary interaction protocols for managing service elasticity operation phase. We focus on interactions between roles and elasticity controllers, but we also support simple interaction among employees, for notifying each other of updates or for delegating responsibility for incidents. We extend SYBL [3], a language for expressing elasticity requirements, to support roles and role-based communication between stakeholders. Based on this, we develop an elasticity Operations Management Platform (eOMP) for cloud services, and we validate its usefulness showing various events encountered for a complex service. eOMP can adapt to various organization structures, and enables service provider employees to interact easily with the elasticity controller, for obtaining a more complex elasticity control. These interactions are real-time, reason for which great care needs to be given to the controller, which should be able to perform normally without human intervention. eOMPs supports managing unexpected situations, by facilitating the collaboration between elasticity controllers and service employees, identifying various types of events occurring during operation phase, and providing mechanisms for solving them.

The paper is organized as follows: Sect. 2 presents the motivation of our work. Section 3 discusses the interactions and roles necessary for elasticity operations. Section 4 describes the design of our platform and the interactions surrounding which the platform is designed. Section 5 presents eOMP case studies, Sect. 6 describes related work, and Sect. 7 concludes the paper and outlines our future work.

2 Motivation

For managing cloud services, a service provider needs to prepare its employees for issues (e.g., errors appearing at system, application or infrastructure level, or change in cloud provider offerings) that could appear in cloud environments, and to adapt their internal processes. Moreover, given cloud environment’s dynamism,

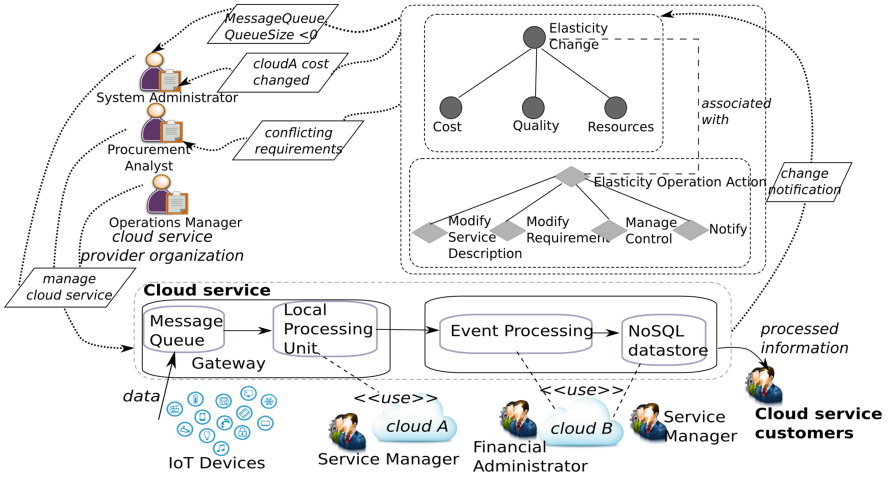


Fig. 1. Motivating scenario

it might be necessary to constantly analyze the service at runtime, to ensure that the service customers receive the expected quality of service. For this, it would be necessary to use an automated elasticity controller (e.g., rSYBL [4]), rapidly reacting to environment changes. The elasticity controller can release from the employees responsibilities, but might cause change in other responsibilities, by including the elasticity controller in the service operation management. Figure 1 shows a complex environment ranging from IoT-specific resources (i.e., sensors, actuators) up to the cloud environment, where information gathered by sensors is stored and processed. In cloud infrastructures, we consider a complex cloud service using various types of software (e.g., NoSQL databases, RabbitMQ), processing data coming from sensors and exposing it to various service customers (e.g., fire department, or building management application). In this paper, we consider cloud service elasticity as being controlled at multiple levels of abstraction, following the cloud service model presented in [4], where the cloud service is composed of units (i.e., elementary parts of the service), which can be grouped into topologies (i.e., semantically connected units).

Focusing on the service provider, we can see that it needs to fulfil various types of requirements (e.g., data placement for IoT device users, or providing expected quality for service customers). While it is obvious that nowadays decision-making solutions should be as much software-based as possible, given the complexity of the setting, there are situations in which it is necessary for the decisions to be taken by real persons playing various *roles* in the organization. Depending on the type of change which has appeared in the service (i.e., following elasticity dimensions: resources, cost, quality [5]), various roles can have different interests or responsibilities in the cloud service control. For instance, whenever the cost of running the platform in the cloud gets high, a *financial administrator* might analyze the overall evolution, and decide whether as a strategic decision it makes sense to use more virtual resources than initially estimated. For this, s/he could either

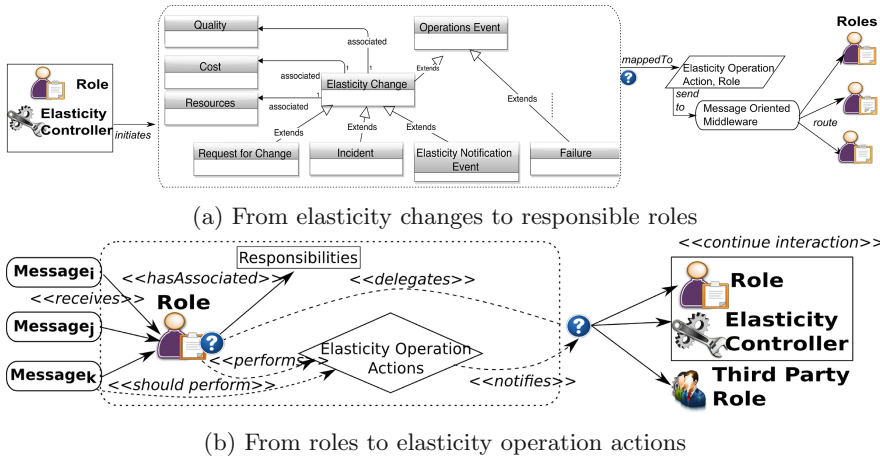


Fig. 2. Role interaction flow

invest more in the platform, or negotiate with *cloud providers* for better prices. Although most times service elasticity is achieved simply by allocating more/-less virtual resources, similar effects might be achieved by changing configurations (e.g., changing the load balancing mechanism). In this case, employees in charge with *configuration management* should know which configuration is being used. The service provider’s goals can also evolve in time, due to varying number or types of users. In our scenario, when adding further data providers for the service, the control requirements need to be modified by the responsible employees, e.g., ensure better data transfer, even though the cost would go over what before was specified as the maximum admissible cost.

Therefore, although an automated solution is necessary for this case, the Level of Automation (LOA) [6] of the elasticity controller should not be of *Full Automation*, but the human (service provider employee) should be included in the control loop, for supporting the cases discussed above, in a *Supervisory Control* mode. Moreover, for achieving this kind of interaction between the elasticity controller and the different types of service provider employees, clear interaction protocols are needed. Since existing solutions mainly focus on full automation [1, 2], we propose using supervisory control for cloud services elasticity, and define interaction protocols for elasticity control. Thus, we introduce a platform easing service provider’s interaction with the elasticity controller, at multiple levels of authority and for multiple elasticity concerns.

3 Analyzing Interactions in Elasticity Operations Management

3.1 Role Interactions

As described in our motivation, the main focus of our work is supporting service providers to achieve better supervised elasticity control. Different service

provider employees are normally in charge with different operations, thus being associated with various roles as part of the organization. We use the term *roles* instead of stakeholders or employee types, to indicate attributions associated to employees/stakeholders, an employee or stakeholder possibly having multiple roles at a time. Moreover, while the roles present in a company are rarely dynamic, the stakeholders/employee types and persons in a company are both volatile and dynamic. Following our motivation scenario, the roles and elasticity controller need to collaborate in order to manage service operation during runtime. As shown in Fig. 2a, roles should be notified by other roles or by the elasticity controller concerning the operation events which occur in relation to the current service. From the operation events, we focus on elasticity changes. We characterize elasticity changes according to the elasticity dimensions (i.e., resources, cost, and quality), and focus on: (i) request for change (RFC), (ii) incident, and (iii) elasticity notification event. The request for change event can be initiated by either an elasticity controller or a role, requesting for changes in properties of the service. The elasticity capabilities (i.e., changes which can be enforced at runtime for modifying service behavior) can incur unexpected behavior in quality, cost or resources, which can indicate an issue in the service configuration or the deployed artifacts. Moreover, service providers are interested in failure events, in order to be able to learn from service behavior and environment changes which produce failures. As shown in Fig. 2b, a role can receive a multitude of messages from other roles or from elasticity controllers. Analyzing them, the role decides whether it can perform needed actions, or if it should delegate to other roles. After analyzing the interaction flow, the next section is focused on analyzing the roles and their responsibilities and interests in elasticity operations management.

3.2 Elasticity Operations and Roles

In the case of elastic cloud services the elasticity controllers play a big role in management, as opposed to ordinary service management roles [7]. Lower level authority roles (i.e., system administrator) have limited responsibilities, supervising and delegating work to the elasticity controller. We focus on designing interaction between elasticity controllers and several roles¹, excluding the roles whose functionalities are replaced by the elasticity controller (e.g., Performance/-Capacity Analyst, Systems Operator, or Capacity Manager).

Table 1 shows the possible elasticity-driven behavior modifications which can be triggered by the elasticity controller, and the roles which might be interested in these types of modifications. Depending on the frequency of modifications, the roles are interested of receiving events more or less often, events being aggregated from a number of modifications, or containing a single modification. For instance, cost-related modifications need to be viewed by finance-related roles, like IT Financial Manager, Procurement Analysis, or Service Manager. Quality-related events (e.g., service part is not healthy, requirements not fulfilled) are of interest for Operations Manager and Service Manager.

¹ <http://www.itsmcommunity.org/downloads/ITIL-Role-Descriptions.pdf>.

4 Elasticity Operations Management Platform

We design our platform (Fig. 3) following the flow described in the previous section, for supporting interactions between roles and elasticity controllers, and even third party roles. The controller sends messages through an *Embedded Queue*, for message locality reasons. From our elasticity Operations Management Platform (eOMP), the *Control Communication* component processes the received notifications. Here we provide a plugin-based mechanism for ensuring that eOMP can be adapted to different elasticity controllers, the only constraint being to map controller notifications to the eOMP model with operation events. The *Control Communication* component processes the operation events, and depending on their types, and depending on the responsibilities of the roles in the *Role Management Component*, maps the controller messages to correct interactions with current roles, and adds them to the queue. For this, we use a *Queue as a Service*, since depending on the number of roles, and possibly in the future, organizations, that we want to support, the scale of the queue and routing complexity can increase. Queue interactions are consumed from the roles' side by the *Interaction processing* component, which directly interacts with the role (e.g., user interface, command line, API-driven communication). For communication between roles, we use a component for collaborative interactions in human-based computing systems (e.g., SmartCOM²).

We designed the platform in such a way that the elasticity controller is agnostic to IT service management processes, or role types. This is beneficial since it enables service providers to choose other controllers, or, for the case of very small company (i.e., one employee), to use the controller without the rest of our platform. The roles and responsibilities can be modified by the eOMP administrator, e.g., by using roles from a different standard.

4.1 Entities of the Interaction

For defining the interactions which may occur for the cloud service elasticity control, we focus on the participants in interactions. We use *organization* to describe an association with a functional structure (e.g., service provider organization). We focus firstly on modeling interactions between the service provider organization and the elasticity controller, and secondly on modeling limited interactions among organizations.

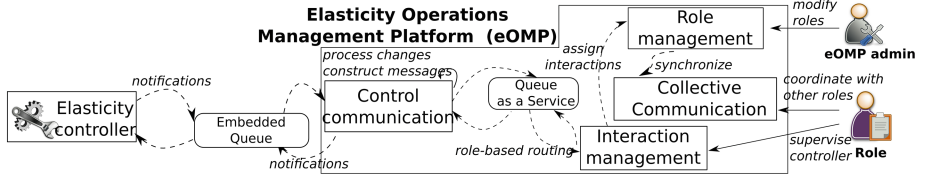
Roles (Eq. 2) are entities which are associated with responsibilities R and authority levels Aut , and which can be assigned to different employees at various points in the organization lifetime. Each organization is defined by a set of role types, which change rarely throughout organization lifetime (e.g., when adopting CMMI³, roles change). *Third party roles* are roles from partner organizations, which are known to the current organization, and are accessible only through an internal role for various actions (e.g., notifications, or contract re-negotiations).

² <https://github.com/tuwiendsg/SmartCom/wiki>.

³ <http://cmminstitute.com/>.

Table 1. Examples of elasticity modifications and roles interested

Modification type	Roles interested
Changes in cost due to scaling/changing the infrastructure/software services which are being used	IT Financial Manager, Service Manager
Changes in cost due to providers change in cost, without change in performance	Procurement Analysis, Service Manager, Operations Manager
Changes in quality due to providers change in quality, without change in cost	Service Manager, Operations Manager
Requirement on cost inflicts degradation of performance	Operations Manager
Configuration change due to change of the workload	Configuration Librarian, System Administrator, Operations Manager
Service part which is not healthy (erroneous, not behaving as expected)	Test Manager, Operations Manager, System Administrator, Incident Analyst
Changes in quality w/o change in workload	Operations Manager, Service Manager
Requirements which are not fulfilled/ are on danger of not being fulfilled by the cloud service elasticity controller	Operations Manager
Data compliance requirements changed by the data provider	Service Manager, Operations Manager

**Fig. 3.** eOMP Design

As we are interested in elasticity control operations management, we focus on *Elasticity Responsibilities* (Eq. 2) related to elasticity dimensions [5].

$$Roles = \{(R, Aut) | R \in Responsibilities, Aut \in [min, max]\} \quad (1)$$

$$Responsibilities = \{x \vee Relations(x, y) | x, y \in \{Cost, Quality, Resources, Error, Analytics\}\} \quad (2)$$

We define a *Message* as being composed of a **header**, and a **body**, the header containing **initiator** and **receiver** related information, while the body contains the **message type**, its **priority** and the **content** of the message. The message content (Eq. 3) can contain suggested interactions, which are nested

interactions suggested by the initiator for the receiver (e.g., an elasticity controller can suggest the Procurement Analyst to re-negotiate the contract with the cloud provider due to cost increase). Using these entities, an *interaction* can be defined as a tuple of *Initiator-Receiver-Message* (Eq. 4), where each of the Initiator and Receiver can be a set of Roles.

$$\begin{aligned} \text{Content} = \{[Cause, SuggestedMeasure]\} \\ Cause \in Requirements \cup ExpectedBehavior \\ SuggestedMeasure \in Interactions \cup Actions \} \end{aligned} \quad (3)$$

$$\begin{aligned} \text{Interaction} = \{[InteractionID, Initiator, Receiver, Message]\} \\ Initiator, Receiver \in Roles, Message \in Messages \}. \end{aligned} \quad (4)$$

4.2 Interaction Protocols for Supervisory Control of Elasticity

As discussed previously, elasticity depends on a large set of variables, both from the IoT, cloud and business world. Elasticity behavior of a service is subject to the business strategy of the service provider, and this can vary with the economic perspectives, and with the market evolution (e.g., the financial manager should decide the strategy in case of financial crisis, and not the elasticity controller). We propose using a *supervisory control mechanism* [6,8], in which any decision of the human overrides any decision of the elasticity controller (i.e., the roles are the outer control loop). For this, we define a set of interaction protocols, based on the entities defined above, for facilitating the communication between roles and elasticity controllers.

Role as Initiator - Bootstrapping Dialogs: The goal of this interaction is to enable roles to initiate dialogs with the elasticity controllers for bootstrapping the elasticity controller (Fig. 4a and b). For *starting elasticity control*, the elasticity controller is sent a **prepare** message, followed by information describing the cloud service is sent one at a time (e.g., service description, custom metrics description), and if each step is successfully achieved, a **Start Control** message is sent. Each call from the role to the dialog starts a complex process on the controller side (e.g., possible elasticity requirement conflicts are solved). For *testing an elasticity capability*, a **Start Test** message is sent for starting the test mode. For instance, the **System Administrator** is able through this dialog to set all needed information for elasticity control (e.g., structure, resources used), and then wait and ensure that each step is successfully completed.

Role as Initiator - Request for Change: The goal of this interaction is to enable the roles to modify expected service behavior during runtime (Table 2). Whenever a role decides that an update is necessary, e.g., due to events signaled by the elasticity controller, the role can modify elasticity requirements, or deployment description (e.g., after a manual re-deployment). For instance, the **Service Manager** can decide to undeploy the service from the cloud environment, and, e.g., keep only an on-premise deployment.

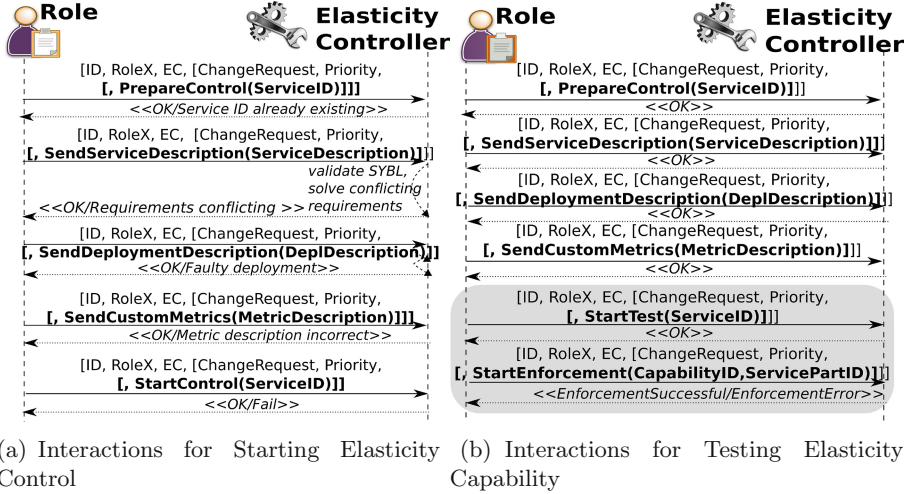


Fig. 4. Interaction dialogs

Table 2. Interactions for requesting modification in control

Interaction type	Interaction details
Undeploy the service	[ID, Role, EC, [RFC, Priority, [Cause, UndeployService (ServiceID)]]]
Replace metric composition rules	[ID, Role, EC, [RFC, Priority, [Cause, ReplaceRules (ServiceID, CompositionRules)]]]
Replace deployment	[ID, Role, EC, [RFC, Priority, [Cause, ReplaceDeployment (ServiceID, DeplDescription)]]]
Replace elasticity requirements	[ID, Role, EC, [RFC, Priority, [Cause, ReplaceRequirements (ServiceID, Requirements)]]]
Pause/Resume control	[ID, Role, EC, [RFC, Priority, [Cause, PauseControl (ServiceID)]]]

Elasticity Controller as Initiator: The goal of this interaction is to enable the elasticity controller to notify appropriate roles on changes in elasticity behavior (Fig. 5). Whenever abnormal changes are observed in the cloud service behavior, the elasticity controller notifies roles, depending on the *Responsibilities*, and the *Authority* which they have associated. For instance, in the case of conflicting requirements which cannot be automatically solved (e.g., response time is expected to be low, while the cloud provider is running in degraded mode), the controller notifies the roles causing the conflict (i.e., $Role_i \dots Role_j$), as well as a *higher authority role* having the responsibilities $\cup_{x=i..j} Responsibilities(Role_x)$.

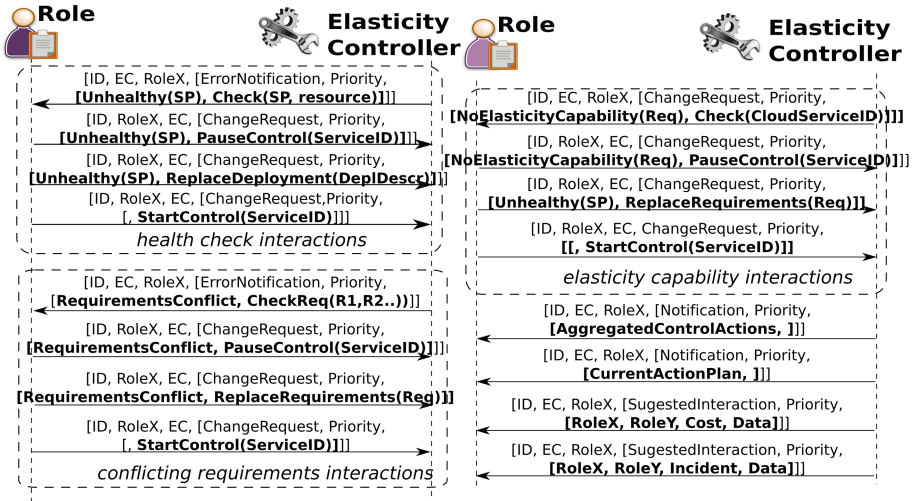


Fig. 5. Elasticity controller bringing the roles into the control loop

4.3 Elasticity Directives-Driven Interactions

For creating custom interactions, we have extended the SYBL [3] elasticity requirements definition language with the new NOTIFY directive, with BNF form described in the Listing 1.1, to be triggered when certain conditions hold. A call of the `notify()` method of the NOTIFY directive maps to the initiation of a new interaction between the elasticity controller and the role mentioned in the directive. An example of such a directive can be `No1: NOTIFY OperationsManager WHEN responseTime > 1.2 s : notify(WARNING, "Response time exceeds 1.2 s")`. Whenever a condition for a notification directive is true, the *Controller Communication* starts an interaction (i.e., translating the aforementioned directive into interaction `[No1, EC, Operations Manager, Notification, "Response time exceeds 1.2 s"]`). However, the frequency of interactions initiated by the *Controller Communication* is adjusted with the interaction aggregation presented in Sect. 4.4.

Listing 1.1. SYBL Notification in Backus Naur Form (BNF)

```

Notification := notificationID:NOTIFY Role WHEN ComplexCondition
               : notify(NotificationType, message)
Role := ROLE(Responsability1, Responsibility2), Role |
        ROLE (Responsability1, Responsibility2) |
        RoleX, Role | RoleX
NotificationType := NOTIFICATION | ERROR | WARNING
    
```

4.4 Interaction Aggregation

For mapping messages, we provide a generic processing mechanism which searches metric patterns associated with responsibilities in the message from the

controller, and creates a new message with the structure described in Sect. 4.1, initiating interactions for the appropriate roles, considering roles' responsibilities. Depending on roles authorities (Eq. 2), interactions are either aggregated or immediately sent to the *Interaction Management* component.

Each role, depending on its responsibilities, receives a different number of messages, or only emergency messages, the amount of messages being inversely proportional with the authority and directly proportional with the responsibilities (e.g., for a maximum Authority of 10, a Service Manager role with an authority of 10 should receive less often messages than the System Administrator with authority 5). Moreover, the nature of the messages should reflect the responsibilities and interests. For this, the *Controller Communication Module* examines messages and identifies metrics of interest for the responsibilities associated with each role. For filtering the interactions initiated, an aggregation function can be defined, selecting the amount of messages to be sent to the roles. The more complex the filtering of the messages, the easier it would be for roles to interact with the elasticity controller. We define in 5 a simple logarithmic *filtering function*, deciding if the aggregation of messages so far should be sent.

$$\begin{aligned}
 f(\text{Role}, Q\text{Interactions}) &= (\log_{\text{auth_max}}(\text{Role.Authority}) * \\
 & \text{THRESHOLD_NOTIFICATION} \leq Q\text{Interactions.size}()) \\
 & \vee (\log_{\text{auth_max}}(\text{Role.Authority}) * \text{THRESHOLD_ERROR} \\
 & \leq \text{MaxPriority}(Q\text{Interactions})). \tag{5}
 \end{aligned}$$

5 Prototype and Experiments

5.1 Prototype

The elasticity Operations Management Platform (eOMP) is implemented as a Java enterprise application, which can be deployed either in the cloud or under service provider's premises. eOMP is open-source and available together with further experiments, details and user guides⁴. The current version integrates with the rSYBL elasticity controller, making use of the notification queue (i.e., embedded queue in the eOMP design, implemented using ActiveMQ⁵) exposing events during runtime. This can be easily extended to other elasticity controllers, by implementing an adapter for receiving and processing events. The service queue is using CloudAMQP⁶, which is managed RabbitMQ⁷ offered as a cloud service. Our Primefaces⁸-based frontend includes dynamically generating diagrams and charts for the cloud service provider employees.

⁴ <http://tuwiendsg.github.io/rSYBL/eOMP>.

⁵ <http://activemq.apache.org>.

⁶ <http://cloudamqp.com>.

⁷ <http://rabbitmq.com>.

⁸ <http://www.primefaces.org/>.

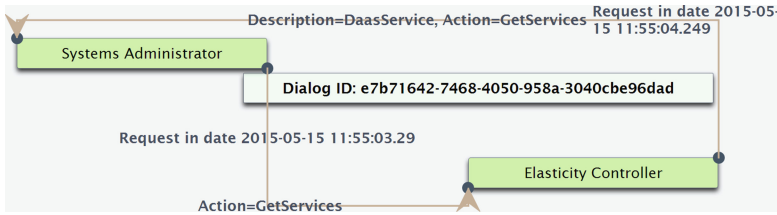
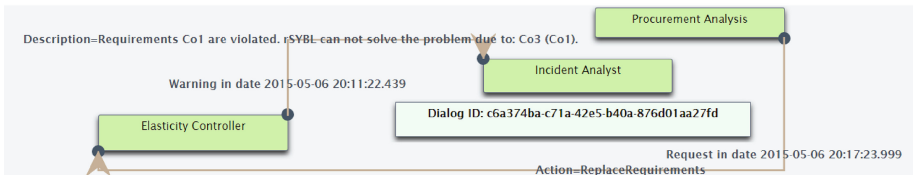


Fig. 6. eOMP snapshot: implicit initial dialog requesting services information

(a) eOMP snapshot: replace requirements



(b) eOMP snapshot: dialog for clarifying requirements

Fig. 7. Conflicting requirements resolution

5.2 Elasticity Operations Management Features

To illustrate eOMP features, we used our pilot application⁹ which consists of: (1) an event processing topology composed of an event processing unit and a load balancer, and (2) a data end topology composed of a data node unit and a data controller unit. We used recordings from a previous run, to which we injected events (i.e., by modifying monitored data for a limited amount of time). We chose this approach instead of real-time injecting faults, since it is more reliable, and our focus is showcasing the eOMP platform, and not the service versatility.

⁹ <https://github.com/tuwiendsg/DaaSM2M>.

Implicit vs. Explicit Interactions. For understanding current service behavior, roles need elasticity controller interactions executed regularly (e.g., every 10 min, or each time the role logs into the platform). We distinguish between two types of interactions from the user/employee perspective: (1) implicit interactions, for getting the necessary data to be displayed to the employee (e.g., dialogs for getting the service description), and (2) explicit interactions, initiated by the eOMP user. Figure 6 shows a dialog from the first type, with the system administrator (one role of the current logged in employee) requesting for initial description information. While without eOMP, the employee would need to manually call them, with eOMP the implicit interactions are already managed when the employee logs on in the platform.

Solving Conflicting Requirements. The controller can encounter a case where no actions are suitable for solving the discovered issues. Figure 7a shows a situation where constraints Co1(“Co1:CONSTRAINT cost < 10\$;”) and Co3(“Co3:CONSTRAINT responseTime < 400ms;”) are conflicting, because of the high workload and the limit on the cost. Since the employee receiving this interaction has more roles associated, s/he decides to replace the requirement from the **Procurement Analyst** role, for being able to increase the limit for the service cost. The interaction dialog from the three roles is shown in Fig. 7b, and consists of two steps: (1) the controller notifies the **Incident Analyst** role that no action is available due to the requirements conflict, (2) the employee uses the **Procurement Analyst** role to modify the cost requirement. Moreover, eOMP uses knowledge on role types and their authorities, avoiding modification conflicts (e.g., with no eOMP, two roles can fix observed issues at the same time). In our case, the role interaction with the elasticity controller is managed by eOMP, the employee being able to choose even a different role from which the issue can be solved better.

Service Health Incidents. Another issue which may occur during service operation is that a service part might be unhealthy (i.e., monitoring metrics have error-like values for an amount of time). With eOMP, all the roles which have incidents as responsibilities receive notifications, but the timing and the amount of notifications is inversely proportional with their authority. Figure 8 shows interactions which are due to **Event Processing Topology** being unhealthy. When the **Operations Manager** (high authority) gets the interaction from the controller, it means that the lower level authority roles have ignored or weren't able to address the situation. Therefore, it delegates the interaction to the **Incident Analyst**. The **Incident Analyst** can try to fix the issue, or can report on the difficulty of the issue. Left side of Fig. 8 shows the actions performed by the **Incident Analyst** (i.e., pause-fix-replace service description). All can be followed by the **Operations Manager**, to make sure the incident is being solved, since these interactions are part of the initial dialog.

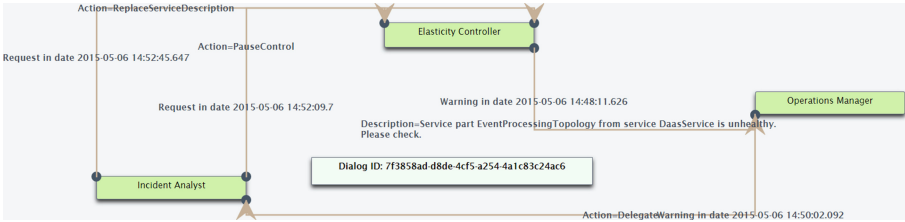


Fig. 8. eOMP snapshot: unhealthy service part dialog

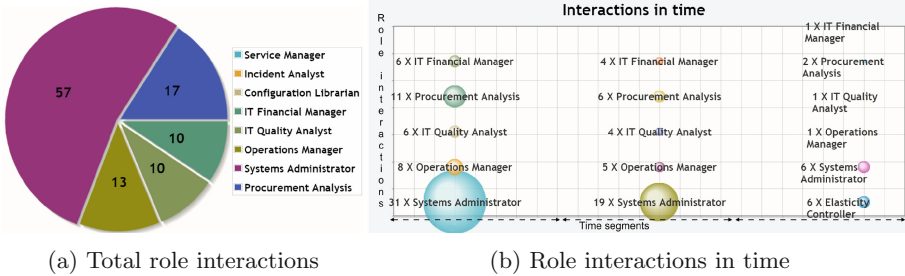


Fig. 9. eOMP snapshots: Statistical information regarding interactions

Dealing with Roles Authorities. Figure 9 shows the number of interactions which occurred over time, undertaken by each role. The amount of interactions varies with the events which occur, with role’s responsibilities in relation with these events, and with role’s authority. We can see that higher level authority roles have less interactions (e.g., Service Manager has no interactions), following the interaction aggregation function in Sect. 4.4.

Thus, eOMP facilitates interactions between service providers and elasticity controllers, and among service provider roles, automating operation tasks and providing support for interaction management based on role’s authority and responsibilities. With eOMP the roles can easily follow the evolution of their elastic service, and the evolution, in time, of incidents, requests for change, or measures taken by the elasticity controller in order to control their service behavior.

6 Related Work

Various standards and processes have been proposed over the years for IT service management. Sallé [9] provides an analysis of the evolution of IT service management over the years, and its evolution towards IT governance. Starting from Information Systems Management Architecture [10], IT management methodologies have evolved towards well-defined standards/best practices of IT service management (e.g., ITIL® [11], BSI ISO 20000 [12], FitSM [13]). Although the focus of this paper is not the management processes adopted by organizations, understanding their internal processes is necessary for being able to support them in their quest for cloud service elasticity. Since the latest reference models used

nowadays in organizations (e.g., [12,14]) are in alignment with ITIL® service management practices, we used these processes and organization roles. However, our design (see Sect. 4) is such that, roles and processes can be modified for accommodating future service management models.

Operation management in cloud computing has been approached mostly from the cloud provider’s perspective [15,16], and little from service provider (i.e., cloud customer) perspective. Bleizeffer et al. [7] propose a set of user roles in cloud systems, having as core roles not only the cloud service provider, but also the cloud service consumer and cloud service creator. The three core roles are expanded into a taxonomy of interconnected user roles, which communicate with each other for delegating responsibilities or gathering information. Demont et al. [17] present an initial proposal of integrating the TOSCA cloud service description standard with ITIL elements. Several commercial solutions enable cloud infrastructure management support, but do not support service operations management at cloud customer’s service level (e.g., Oracle Enterprise Manager¹⁰, BMC Cloud Operations Management¹¹). Liu et al. [18] propose an incident diagnosis approach based on incident relationships, using co-occurring and re-occurring incidents for performing root cause analysis. Munteanu et al. [19] propose an architectural approach for cloud incident management, including incident lifecycle management, event and incident detection, incident classification and recovery and root cause analysis.

In contrast with above presented work, we focus on the elasticity aspect of service operations management in the cloud, characterizing the relevant properties and interactions. Moreover, we emphasize the importance of supervisory control for the cloud, and introduce service provider employees as first-class entities in the control loops.

7 Conclusions and Future Work

Operation management for services elasticity becomes increasingly more complex when the services rely on several other third-party services deployed in multiple cloud environments. We have proposed a set of interaction protocols for managing elasticity operations of cloud services, which take into consideration service provider roles as first class entities in the service elasticity control. We introduced the eOMP platform, which allows service provider employees to manage the cloud service operations related with elasticity, interacting with the elasticity controller and other employees of the service provider.

As future work, we are studying multi-organizational interactions for cloud services. For this case, dialogs are much more complex, since we need to model the various types of information necessary in the communication, and we need mechanisms to support and track inter-organization interactions. Moreover, organizations could follow different standards, and offer different access points (e.g., a cloud provider might expose IT Financial Administrator role for re-negotiating contracts, and Operations Management for handling QoS issues).

¹⁰ <http://www.oracle.com/technetwork/oem/enterprise-manager>.

¹¹ <http://www.bmc.com/it-solutions/cloud-operations-management.html>.

References

1. Mao, M., Humphrey, M.: Scaling and scheduling to maximize application performance within budget constraints in cloud workflows. In: IEEE 27th International Symposium on Parallel Distributed Processing (IPDPS), pp. 67–78 (2013)
2. Jiang, J., Lu, J., Zhang, G., Long, G.: Optimal cloud resource auto-scaling for web applications. In: 2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 58–65 (2013)
3. Copil, G., Moldovan, D., Truong, H.L., Dustdar, S.: Sybl: an extensible language for controlling elasticity in cloud applications. In: International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 112–119. IEEE/ACM (2013)
4. Copil, G., Moldovan, D., Truong, H.-L., Dustdar, S.: Multi-level elasticity control of cloud services. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 429–436. Springer, Heidelberg (2013)
5. Dustdar, S., Guo, Y., Satzger, B., Truong, H.L.: Principles of elastic processes. *IEEE Internet Comput.* **15**(5), 66–71 (2011)
6. Endsley, M.R., Kaber, D.B.: Level of automation effects on performance, situation awareness and workload in a dynamic control task. *Ergonomics* **42**, 462–492 (1999)
7. Bleizeffer, T., Calcaterra, J., Nair, D., Rendahl, R., Schmidt-Wesche, B., Sohn, P.: Description and application of core cloud user roles. In: Proceedings of the 5th ACM Symposium on Computer Human Interaction for Management of Information Technology. CHIMIT 2011, pp. 2:1–2:9. ACM, New York (2011)
8. Sheridan, T.B.: Adaptive automation, level of automation, allocation authority, supervisory control, and adaptive control: distinctions and modes of adaptation. *IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum.* **41**(4), 662–667 (2011)
9. Sallé, M.: It service management and it governance: review, comparative analysis and their impact on utility computing. Hewlett-Packard Company, pp. 8–17 (2004)
10. Van Schaik, E.A.: A Management System for the Information Business: Organizational Analysis. Prentice-Hall Inc., Upper Saddle River (1985)
11. Arraj, V.: Itil®: The Basics. Buckinghamshire, UK (2010)
12. BSI Group: ISO/IEC 20000-Information technology-Service management. http://www.iso.org/iso/publication_item.htm?pid=PUB200013
13. FedSM: FitSM: Standards for IT Service Management. <http://www.fedsm.eu>
14. HP: The HP IT Service Management (ITSM) Reference Model. ftp://ftp.hp.com/pub/services/itsm/info/itsm_rmwp.pdf
15. Zhan, H.J., Zhang, W.: The operation and maintenance management system of the cloud computing data center based on ITIL. In: Jeong, H.Y., Obaidat, M.S., Yen, N.Y., Park, J.J.J.H. (eds.) Advanced in Computer Science and Its Applications. LNEE, vol. 279, pp. 1103–1108. Springer, Heidelberg (2014)
16. IBM: Integrated service management and cloud computing: More than just technology best friends. https://www.ibm.com/ibm/files/E955200R99025N70/5Integrated_service_management_and_cloud_computing_644KB.pdf
17. Demont, C., Breitenbücher, U., Kopp, O., Leymann, F., Wettinger, J.: Towards integrating tosa and itil. In: ZEUS, Citeseer, pp. 28–31 (2013)
18. Liu, R., Lee, J.: IT incident management by analyzing incident relations. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) ICSOC 2012. LNCS, vol. 7636, pp. 631–638. Springer, Heidelberg (2012)
19. Munteanu, V., Edmonds, A., Bohnert, T., Fortis, T.F.: Cloud incident management, challenges, research directions, and architectural approach. In: International Conference on Utility and Cloud Computing, pp. 786–791. IEEE/ACM (2014)

rSLA: Monitoring SLAs in Dynamic Service Environments

Heiko Ludwig¹(✉), Katerina Stamou¹, Mohamed Mohamed¹,
Nagapramod Mandagere¹, Bryan Langston¹, Gabriel Alatorre¹,
Hiroaki Nakamura¹, Obinna Anya¹, and Alexander Keller²

¹ IBM Research - Almaden, San Jose, CA, USA
{hludwig, stamou, mmohamed, pramod, bryanlan, galatorr,
obanya}@us.ibm.com, hnakamur@jp.ibm.com

² IBM Global Technology Services, Chicago, IL, USA
alexk@us.ibm.com

Abstract. Today's application environments combine Cloud and on-premise infrastructure, as well as platforms and services from different providers to enable quick development and delivery of solutions to their intended users. The ability to use Cloud platforms to stand up applications in a short time frame, the wide availability of Web services, and the application of a continuous deployment model has led to very dynamic application environments. In those application environments, managing quality of service has become more important. The more external service vendors are involved the less control an application owner has and must rely on Service Level Agreements (SLAs). However, SLA management is becoming more difficult. Services from different vendors expose different instrumentation. In addition, the increasing dynamism of application environments entails that the speed of SLA monitoring set up must match the speed of changes to the application environment.

This paper proposes the rSLA service and language that is both flexible enough to instrument virtually any environment and agile enough to scale and update SLA management as needed. Using rSLA the time of setting up SLA compliance monitoring of application environments involving infrastructure, platform, and application services can be significantly reduced.

Keywords: Service Level Agreement · Cloud Computing · PaaS · Monitoring · Reporting

1 Introduction

Cost, availability and on-demand scaling have accelerated the adoption of different application deployment models. Today's application environments combine Cloud and on-premise infrastructure as well as platforms and services from different providers to enable the quick development and delivery of solutions to their intended users. For example, a Web or mobile application of an online

merchant might be deployed on a cloud-based Platform-as-a-Service (PaaS), use persistence services within the platform, run ID and login management through a third-party Web service, and be monitored by various in-house logistics systems.

The ability to use Cloud platforms to stand up applications in a short time frame, the - now - wide availability of services, and the application of a continuous deployment model has led to a much more dynamic application environment, changing at high velocity.

Managing quality of service has become more important in the context of the Cloud and the prolific use of micro-services. The more external service vendors are involved the less control an application owner has over the quality of the delivery of this service and must rely on the quality commitments of his or her vendors in the form of Service Level Agreements (SLAs).

SLAs are widely used in the context of corporate Information Technology (IT) outsourcing. Practically every outsourcing customer requires his or her vendors to commit to SLAs and holds vendors accountable with financial penalties or rewards for achieving objectives. Objectives typically relate to the performance of systems and services, their availability, and the performance of service processes such as provisioning servers and responding to help desk tickets. Many vendors sell SLA management systems, often as part of service management suites (e.g., IBM Integrated Service Management/ISM) or as Software-as-a-Service (e.g., ServiceNow). In enterprise practice setting up SLA management for a particular domain and monitoring compliance on an ongoing basis requires substantial integration projects to collect metrics in a data warehouse. Aggregating and adjudicating SLA compliance likewise often requires substantial manual work or ad-hoc scripting.

While this is viable for large outsourcing contracts and all their intricate details this is not a good match for today's dynamic application environments using Clouds and services from multiple vendors. One key problem is the heterogeneity of interfaces. Services from different vendors have different instrumentation and use different service management systems making it difficult to collect and aggregate performance data for service level objective evaluation. While system vendor heterogeneity is certainly an issue within one enterprise it is easier to avoid and system level standards such as the Common Information Model (CIM) of the Distributed Management Task Force (DMTF) help management systems to interact with servers, network components and the like. Management APIs on a Cloud and service level have not yet undergone such widely accepted standardization although some schemes have been proposed [6].

In addition to heterogeneity, the increasing dynamism of application environments entails that SLA monitoring must be set up at the speed of change of the application environment. Continuous deployment, or DevOps, enables constant changes to applications and the binding to new services. Cloud infrastructure and - in particular - platform services enable the deployment of new applications on very short notice. Organizations can respond rapidly to novel needs and change their application environment at a fast pace.

To provide effective SLA management in a Cloud environment, the SLA management system must be able to set up the monitoring of customer-specific SLA terms against a heterogeneous set of service instrumentation in a short amount of time.

This paper proposes the rSLA service, which is both flexible enough to instrument virtually any environment and agile enough to scale and update SLA management as needed. Given an SLA expressed using a formal SLA specification, rSLA sets up the monitoring infrastructure and starts monitoring compliance. Using rSLA the time of setting up SLA compliance monitoring of application environments involving infrastructure, platform, and application services can be significantly reduced and aligned with a typical DevOps life-cycle.

The remainder of this paper is organized as follows: The next section discusses related work. Subsequently, we give an overview of the approach, followed by a discussion of the rSLA language. After that we outline architecture and execution model, including the implementation. In this section we also show in a case study how we applied our approach to a real customer environment. Finally, we summarize and conclude.

2 Related Work

QoS has been an important part of IT Service management for a long time. As a result, a significant body of work has focused on different aspects of QoS - from measurement to enforcement.

In addition to typical IT service SLA management products mentioned in the introduction, many providers of Cloud and networking services do provide specific service performance/availability information on their Web sites or through APIs. This is often the basis of tiered service offerings, with better quality demanding a higher price. For instance, Amazon provides a generic monthly uptime SLO of 99.95 % for their compute resources and performance SLO of 2 k IOPS for EBS SSD volumes at the point of writing of this paper. [2] provides a survey on current service level provisioning practices and compares the SLAs of five public cloud service providers. While this typically entails some kind of availability/performance guarantees it is not a custom SLA, tailored to a client's business needs. This one size fits all policy is often too restrictive.

An important part of a solution to automate SLA management is a formal, machine-interpretable representation of the SLA. Several specifications have been proposed in the Web service and Grid context such as the Web Service Level Agreement language (WSLA) [10], the Web Services Offer Language (WSOL) [17] and Web Service Agreement (WS-Agreement) standard of the Open Grid Forum [1]. WSLA is designed to capture SLAs for web services and is used, among other things, to facilitate automatic service configuration, e.g., to configure load balancers of clustered Web servers. WS-Agreement inherits language characteristics of the WSLA specification and has been used in numerous initiatives [3, 16]. SLA representations were also used for on-demand service provisioning [5, 9] and automated service composition [18].

These uses require the automatic set up of SLA compliance monitoring for the systems they are applied to, e.g., Web services. While reading metrics at specific points of a distributed cluster could be described in some representations such as WSLA there was no prescribed way to address the heterogeneous instrumentation found in today's Cloud scenarios. It was not really necessary because the systems instrumented were assumed to be Web servers, Grid schedulers, etc. A number of approaches that use SLAs as a key element of compliance evaluation, such as [4, 13, 14], have been proposed for application monitoring in dynamic systems. Like these authors, we focus on comprehensive QoS monitoring of SLAs. However, our approach includes a flexible and scalable monitoring mechanism and action framework required for quick setup of SLA monitoring in today's heterogeneous and dynamic cloud environment.

Recent work has investigated SLAs for Cloud deployments. In [7] Kouki et al. propose the Cloud Service Level Agreement language (CSLA) that enables the definition of SLAs for any type of cloud service. CSLA is intended for dynamic service provisioning environments and is based on the Open Cloud Computing Interface (OCCI) [15] and the Cloud Computing Reference Architecture of the National Institute of Standards and Technology (NIST) [8]. While these proposals define some common metrics and SLA parameters to set individually they reflect more an advertised QoS than a commitment to an individual customer.

In contrast, our work focuses on building a light weight, extensible system for SLA management with the goal of rapid deployment and adaptability to ever changing cloud landscape and customizability to different client business requirements.

3 Overview of the Approach

The key objective of our SLA compliance monitoring system presented in this paper is the fast setup of monitoring an SLA in an environment in which the subject of monitoring is evolving quickly and services expose heterogeneous measurement interfaces. This section provides an overview of our approach how to address all of those issues at the same time. The first subsection discusses the system model outlining how the rSLA service interacts with interfaces of a heterogeneous environment. The second part describes the major elements of an SLA document's content.

3.1 System Model

Cost, availability and on-demand scaling have accelerated customer adoption of different application deployment models - on-premise Cloud, private Cloud, public Cloud or hybrid Clouds, in addition to their traditional dedicated environments. Depending on the services an application depends on, the choice of platform varies. A single application could potentially be reliant on multiple Clouds for the set of services it consumes. From a customer or tenant perspective, monitoring service levels that their workloads are receiving often becomes a complex task of aggregating information across multiple Cloud providers, each

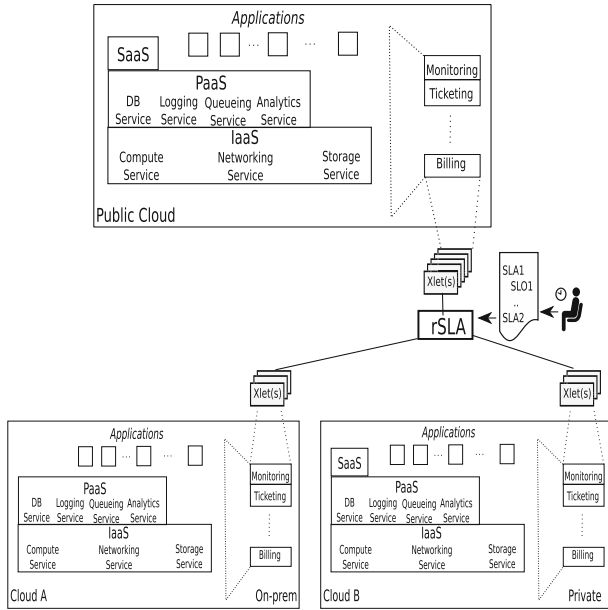


Fig. 1. rSLA system overview

using different proprietary monitoring and management stacks. Furthermore, the DevOps transition increases the rate of change of applications and consequently their bindings to services.

Figure 1 shows an overview of the proposed rSLA system model. The rSLA framework is made up of three main components:

1. a language for administrators or service providers to express service level agreements;
2. a set of Xlets - lightweight, dynamically bound adapters; Xlets abstract the heterogeneity of service management interfaces to rSLA, both for the reading of metrics as well as the acting on the result of SLA evaluation; and
3. a modular SLA evaluation service; this service interprets the rSLA document. It performs the reading of measurements through monitoring Xlets as defined in the document, aggregates metrics, evaluates compliance and then notifies stakeholders.

A formal language for SLAs is the key to both custom SLAs on a case-by-case basis and fast deployment of the SLA. The Xlet approach provides the level of abstraction and indirection to intermediate between service interface heterogeneity and the need of the SLA evaluation service to read metrics in a homogeneous way.

The rSLA evaluation itself can be deployed in any of the Cloud deployment models - on-premise Cloud, private Cloud, public Cloud or hybrid Clouds. Further sections describe the language concepts, the execution model, and implementation. The case study later in the paper discusses some Xlets we wrote for a particular Cloud service.

3.2 SLA Model

An rSLA document describes how metrics are to be obtained from instrumentation and how they are to be aggregated. Based on those metrics Service Level Objectives (SLOs) can be defined and it can be specified how to proceed if those SLOs are met or violated. Figure 2 illustrates the main concepts of the proposed approach using a simple example.

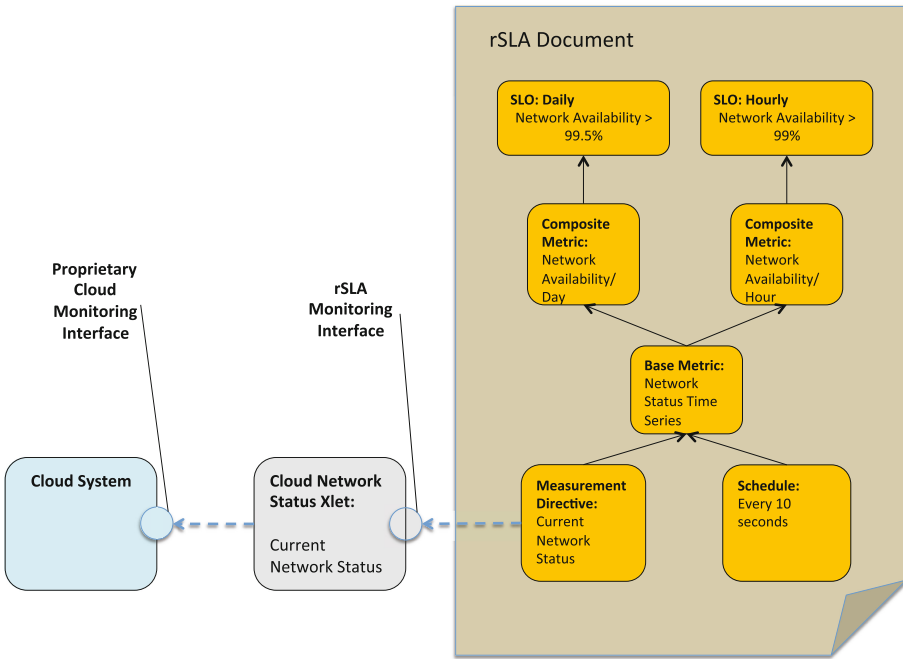


Fig. 2. rSLA concepts overview

This example describes an SLA defining networking guarantees. The customer is interested in a daily guarantee, which needs to exceed 99.5 % availability, and an hourly one, which needs to be only 99 %. The current status of the network at any given point in time can be obtained at the proprietary interface of the Cloud provider with whom the customer enters the SLA. There is an Xlet exposing an rSLA monitoring interface that can be requested to provide network status in an rSLA standard way and which in turns requests this status from the proprietary Cloud management interface.

When expressing the rSLA document the author of this document has to describe how to translate what can be read by instrumentation into a high-level metric on which the customer wants to have a commitment. In addition, the actual SLO must be articulated. This gives us our main set of concepts for rSLA:

The foundation of measurement are *base metrics*. Base metrics describe how metrics are to be obtained from instrumentation and how frequently. For this purpose a base metric has a measurement directive and a schedule.

The *measurement directive* describes how a specific metric can be obtained through an Xlet. It points to an Xlet, the specific service entity, and the metric to be obtained. For example, our networking Xlet for this Cloud provider may provide status for an external access network as well as the internal network, connecting the different virtual machines of this customer (service entities). Multiple metrics may be available for those entities such as availability and latency. The measurement directive will point to the specific metric of a specific entity.

The *schedule* describes the frequency at which measurements are taken. The periodicity can reflect the customer's needs. For our networking example this time frame will typically be a sampling interval in a range of seconds to minutes.

Composite metrics aggregate values of base metrics and other composite metrics. The aggregations are described using expressions over values from the metrics it depends on. The phrasing of the expression must be type-compatible with the types of the input metrics. Oftentimes, expressions involve aggregations of time series, e.g., to averages over a time period, or aggregate different metrics such as the network availability of multiple network segments. In our example, two composite metrics depend on the same base metric, one aggregating network state in the window of the last hour, one in the window of a day. While the basic approach to metric aggregation is simple, we often find quite complex functions excluding some values and performing complex stochastic operations. This puts significant requirements on the richness of expressions the language provides.

Service Level Objectives define the commitment of a service provider to its customer. This is defined in a Boolean expression over metric values. Oftentimes, commitments are bounded by a precondition. For example, a response time guarantee for a REST call is only given if the request rate to the service is less than a certain number of calls per minute. This is commonly used to scope the applicability of the SLO. SLOs can also be associated with a schedule that defines when to evaluate the precondition and objective expressions. Alternatively, it can be evaluated each time input metrics have new values. This might lead to a significant computation effort when large numbers of metrics are sampled at high frequency.

Finally, though not shown in the figure, *notifications* define how external services are to be informed about the state of SLOs and metrics. The rSLA language provides notifications as event-condition-action rules that describe when notifications should take place, according to a schedule or when a new evaluation is available; the specific condition, e.g., upon violation; and how to notify. For this “how” part, the action, we fall back to our Xlet concept. Notification Xlets provide a homogeneous interface abstracting from various possible recipient interfaces. Notification directives describe how to connect to those Xlets. Notification statements also include a description which information is to be passed on, the marshalling of the action.

Based on this small set of concepts SLA authors can express a large variety of SLAs. Measurement directives connect the rSLA language to the systems model; composite metrics can express custom metrics aggregation to the level needed by the client; SLOs express the commitment; notifications define how to actively communicate the SLO status and other information to recipient services.

4 rSLA Language

The rSLA language allows an SLA author to express the concepts outlined in the previous section. The specific language design is key to the consumability of the language and the usability of the approach as a whole. Prior to diving into the specifics of our language elements we want to briefly discuss some issues and guiding principles.

4.1 Design Considerations

Many of past and current approaches to representing SLAs such as WSLA, WS-Agreement, WSOL and CSLA choose XML as a substrate. While some of those languages have enjoyed success in systems deployment or as basis for further research work none has seen wide-spread adoption in industry at this point. While not having systematically studied adoption, the authors of this paper have been involved in some of those efforts and received feedback as to why or why not to use one language or another. A key item of feedback from practitioners was that XML is hard to read. While there is always the opportunity to provide advanced editors to not expose authors to the actual representation many professionals actually prefer writing system-level scripts.

When considering who actually writes rSLA documents it turns out that this would often be administrators with significant experience in scripting. The rSLA syntax is designed to resemble other languages that administrators use on a daily basis. Many domain specific languages have seen acceptance lately, e.g. Chef (www.chef.io). rSLA aims at providing a similar experience to its authors.

Another issue with early efforts such as WSLA is the limitation of its expression language, or its complete absence in the normative specification, in case of WS-Agreement. While WSLA has an extensible set of functions, a practitioner would actually have to change its evaluation engine to extend it. This is not practical. WS-Agreement suggests to use the expression language of your choice, again requiring an addition to a runtime system. From a practitioner's perspective they are incomplete. The rSLA language is meant to encompass a wide set of functions that can be used in expressions. In addition, the language design enables to define extension to the standard language expression scope as part of the SLA document itself.

To achieve those objectives the rSLA language is designed as a Domain Specific Language (DSL) on the basis of Ruby - hence the "r" in rSLA. Ruby, as a substrate language, has some characteristics that makes it suitable as a

basis of a DSL, in particular its scripting approach and its easy access to meta-interpretation. Many DSLs such as Chef are also based on Ruby and administrators often have encountered those before. Other languages and their interpreters might be equally suitable from a technical requirements perspective.

4.2 rSLA Language Elements

This section discusses the rSLA language elements in an overview and example manner. We provide a small rSLA document that specifies three rSLA objects: an SLA, a base metric and a service level objective. The SLA can be read by an rSLA engine in a cloud runtime environment. A full specification of the language is beyond the scope of an individual paper of this format. The rSLA language documentation describes in detail the language structure, the use of rSLA statements, and its production rules [11].

Listings 1.1 and 1.2 illustrate example statements for the creation of an SLA and a base metric and, respectively, of an SLO object. A deployed rSLA service can read and process these statements in a runtime environment. An rSLA document must have exactly one *sla* statement. The *sla* statement defines the parties to the SLA.

```

1  sla do
2    tenant "ExampleClient"
3    provider "ExampleProvider"
4  end
5
6  basemetric do
7    name "bareMetalProvisioning"
8    unit "seconds"
9    type "event_set integer"
10
11   measurementdirective do
12     entity "/process/baremetal_provisioning"
13     type "event_set integer"
14     source "http://provisioningxlet.stage1.mybluemix.net/
           process/baremetal_provisioning/time"
15   end
16
17   schedule do
18     frequency "1"
19     unit "m"
20     method "every"
21   end
22 end

```

Listing 1.1. rSLA SLA (lines 1–4) and basemetric (lines 6–19) statements

The base metric definition includes its *measurementdirective* and may include a *schedule* if values are received according to a schedule. Its attributes are *name*, *unit* and *type*.

The type of this particular example is an *event_set* of *integers*. Types of metrics can be either simple types or one of two complex types: *time_series* and

event_set. An event set is a set of values collected over time. Typical examples include provisioning events. In a given time interval any number of provisioning processes may have occurred. If a provider wants to give a guarantee over the time it takes to provision, say for 90 % of the processes, we need a complex type able to accommodate this set of values representing the time it took. Time series, on the other hand, are equidistant readings of values.

The measurement directive describes where to read the base metric values. It describes the entity in question, its type, and the source. In this specific case the source is a fully articulated URL. Based on the detailed metric name and the entity specification a reference to an Xlet name could have sufficed as well.

The schedule follows the syntax of the Rufus scheduler [12], which is sufficient for our purposes.

Another required element of an SLA is the SLO. An example of such an SLO statement is illustrated by Listing 1.2.

```

1 slo do
2     name "bareMetalProvisioning"
3     precondition "bmtotalProvisionsNumber.value≤100"
4     objective "bmGoodProvisionsNumber.value\
        bmtotalProvisionsNumber.value≥0.9"
5
6     schedule do
7         frequency "60"
8         unit "m"
9         method "every"
10    end
11 end

```

Listing 1.2. rSLA SLO creation script

The SLO definition in Listing 1.2 provides an example of an SLO statement in the rSLA language. The *slo* has a name to refer to, a schedule according to which it is evaluated and two expressions: the *precondition* defines the bounding condition of the SLO while the *objective* defines what must be achieved.

Upon SLO evaluation, the rSLA engine will first evaluate the precondition block. If the logical outcome from the execution of the precondition block is false, the SLO is met. In our case, if the total number of bare metal servers provisioned in the past hour is less or equal than hundred, the SLO applies. If it is larger, it doesn't - and no violation occurs. In case the precondition is true or if there is no precondition block, the rSLA runtime evaluates the objective block. If the logical outcome from processing the statements in the objective block is true the SLO is healthy. Otherwise the SLO evaluation indicates a violation.

Both base metric and SLOs enable a user to define schedules for the measurement and respectively the evaluation of such rSLA instances. Schedule details are passed to a scheduler service that instantiates schedule objects and coordinates their processing.

Both precondition and objective are expressions following Ruby syntax. Values of metrics, both base metrics and composite metrics, are referred to by

their name “.” value. This can refer to a complex type or a simple type. Ruby expressions resolving to a Boolean type are valid. This enables authors to use any available Ruby function, providing the full expressiveness of the substrate scripting language.

Not illustrated for space reasons is the *composite_metric* statement. It has name, unit and type like the *base_metric* but rather than a measurement directive it has an expression as well, which has to yield a result according to its type. Equally, all type-compatible Ruby expressions are valid.

This syntax allows administrators and others of similar skill to define SLAs in a scripting-like DSL. Using an existing scripting language as substrate, in our case Ruby, enables us to use a full scope of expressions.

5 rSLA Runtime Architecture and Implementation

The rSLA Service is designed to be run either by a service provider, a customer, or a third party such as a service integrator. The following section explains the components constituting the rSLA service and its interaction with xlets.

5.1 rSLA Service

The rSLA service is a Ruby application, a given because it is a Ruby DSL. rSLA offers different REST interfaces that allow the management of the life cycle of an SLA described using our DSL.

The rSLA service exposes a set of life cycle management functions to its user, in particular to create, activate, deactivate and remove SLAs. At the reception of a new SLA, the rSLA Service interprets the file and creates a set of objects. The new objects are persisted in a Cloudant no-SQL database service using a CouchRest data model.

On the *activation* of an SLA, the rSLA Service orchestrates all the needed operations to activate and manage the SLA life cycle. It starts by scheduling data collection for base metrics with the rSLA scheduler service, which is a companion service to the rSLA application itself. Based on the defined schedules, the scheduler triggers rSLA Service interfaces exposed to the scheduler associated with metrics. The rSLA service then invokes the monitoring Xlets to collect new observations for the related base metrics. The observations received from Xlets are persisted in Cloudant. Observations are JSON structures returned by monitoring Xlets that contain the metric value requested, a time stamp and an *extras* object that contains more information useful for further diagnostics. For example, in case of our provisioning processes, in addition to the provisioning time as the value of the metric it may be interesting to know which particular server has been provisioned if a provisioning process takes longer than expected.

Similarly, according to the schedule of an SLO, the Scheduler triggers an rSLA Service interface for SLO evaluation. The rSLA service then retrieves the data related to the specific SLO, reifies the corresponding metrics as Ruby

objects and then evaluates the expressions of the SLO, precondition and objective. As an optimization step, this evaluation may use the map-reduce functions offered by Cloudant to delegate possible parallel processings to Cloudant and benefit from its efficiency.

Afterwards, the rSLA Service generates JSON notifications representing the results of the evaluation. These notifications are sent to the Notification Xlet for formatting and reporting to the client.

5.2 rSLA Xlets

As described previously, Xlets offer a standard interface for monitoring and notification as a generic REST API, abstracting from the heterogeneity of different service interfaces. In our architecture, Xlets are services on Bluemix. An Xlet is customized according to its role in the overall system. As shown in Fig. 3, each Xlet provides three interfaces:

- *CFBrokerInterface*: Since the Xlets are provided as services on Bluemix, they need to offer this generic interface that describes exactly how to provision the service, how to unprovision it, how to bind the service to a given application and how to unbind it.
- *ConfigurationInterface*: In order to ensure multi-tenancy and customization of an Xlet, it must offer an interface to configure its tenancy. This interface could offer other functionalities of customization, e.g., to configure the access credentials for Cloud resources.
- *RuntimeInterface*: This interface provides the main business of the xlet. It describes the specific functionalities to be offered by the application instance (e.g., monitoring services, reporting services).

All Xlets observe the same architecture but differ in their implementations from one use case to another. The runtime interface of the monitoring Xlets are in line with the DMTF Cloud Infrastructure Management Model [6]. They allow collecting monitoring data for a specific type of resources with different granularities. For example, Fig. 3 shows a SoftLayer specific Xlet. The SLXlet allows to get monitoring data of SoftLayer-provisioned servers for a given account. The account credentials are passed to the Xlet in the configuration phase through the Configuration interface. Afterwards, the runtime interface of the Xlet can be used to get the list of servers, the list of metrics for a given server, or the value for a given metric for a specific server.

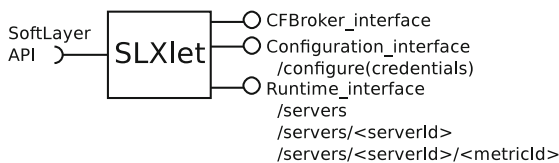


Fig. 3. Xlet interface design

Using a PaaS as Xlet platform, in our case the Bluemix implementation of Cloud Foundry, has advantageous characteristics: *scalability* is inherited from the scalability of the Bluemix environment. Since the Xlet can be provisioned as an application or a service within Bluemix, it is easy to scale it horizontally to cope with the work load by adding or removing new instances. All Xlets have a common and generic core code that often allow the easy *reusability* with minor modifications for specific use cases. *Managing* Xlets is handled to Bluemix, the management here includes provisioning, deprovisioning, binding and unbinding Xlets to the rSLA service. Xlets can be integrated *flexibly* using Bluemix services and can be provisioned using different plans, e.g., shared or dedicated. The rSLA architecture itself does not depend on a specific PaaS code base or service.

5.3 Case Study

We have conducted a pilot of the rSLA service to support the management of Cloud services used by an existing customer. In one agreement, the customer migrated a workload from a customer-owned, on-premise data center environment to the IBM Softlayer Infrastructure-as-a-Service. Along with the move, the customer required the monitoring of seven custom SLOs that had never been offered previously by the service provider in the Cloud in this form.

Each of the seven SLOs consist of one base metric and one service level objective, with multiple composite metrics used to aggregate to level needed by the SLOs. The rSLA document has been defined based on the written agreement with the customer and discussions with the client. The rSLA was submitted to and executed by the rSLA service to activate and initiate the measurement of the involved base metrics. The rSLA service monitors, measures, evaluates and reports the service level status of the seven involved SLOs on a daily basis, resulting in about 100 MB/month in observation data.

We developed three different Xlets to monitor various aspects of the service such as network status and provisioning process times, as well as one Xlet for email notification. Overall, it took 3 weeks from first contact with the project team to the start of monitoring. This included the development of the specific Xlets, the definition of the SLA document, and obtaining the access keys to the client's Cloud service API. This is much faster than a traditional integration to an enterprise SLA management system. In a future scenario, these Xlets could be reused for another client of Softlayer requesting an SLA related to the same base metrics. The remaining effort only comprises the writing of the SLA document and obtaining the API keys for monitoring Xlet configuration, which may be as little as a number of hours or a day. Over time, having accrued a variety of Xlets for various service interfaces, SLA monitoring for popular services can be set up just based on an rSLA document.

6 Summary and Conclusions

Today's cloud-based application environments enable their users to deploy and change applications constantly, binding to different services and platforms on

short notice. Traditional enterprise SLA management typically requires a complicated setup process lagging far behind the application life cycles of a DevOps environment. Current industry practice to Cloud SLA management often fails to take into account specific customer needs. Existing approaches for Web and Grid services often fail to deal with interface heterogeneity in an effective way and have a syntax that is often perceived to be cumbersome by practitioners.

The rSLA approach presented in this paper addresses the efficient specification of SLAs in a formal language and, at the same time, uses the Xlet architecture abstractions to overcome issues of heterogeneity. While reusing some existing concepts such as metrics and SLOs, rSLA makes a number of significant contributions to meet the objective of fast SLA deployment in a Cloud environment: Xlets provide a standard way to refer to and use diverse interfaces. The concept of rSLA measurement directives ties base metrics to the way metrics can be obtained through Xlets, thereby enabling references to metrics from various interfaces in the language. The design of the rSLA language as a Ruby DSL provides access to a full expression language in a way that is familiar to many target users such as administrators using Chef.

The approach has been implemented on Bluemix and tried out in a pilot, monitoring IaaS-related SLOs. This has been accomplished much faster than using an enterprise SLA management system and we were able to deal with a previously unknown management API, which Web service-oriented systems such as WSLA cannot. In addition, the resulting SLAs are actually legible. While we have obtained first results of our approach presented in this paper further work will be required addressing expressiveness of different scenarios, performance, and user acceptance.

The current implementation still has some limitations: it does not allow a user to specify his own measurement mechanisms and xlets are bound statically. We expect to address some of those issues in future work.

References

1. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement), September 2005. <http://mailman.ogf.org/documents/GFD.107.pdf>
2. Baset, S.A.: Cloud slas: present and future. *SIGOPS Oper. Syst. Rev.* **46**(2), 57–66 (2012)
3. Butler, J., Lambea, J., Nolan, M., Theilmann, W., Torelli, F., Yahyapour, R., Chiasera, A., Pistore, M.: SLAs empowering services in the future internet. In: Domingue, J. (ed.) *The Future Internet*. LNCS, vol. 6656, pp. 327–338. Springer, Heidelberg (2011)
4. Comuzzi, M., Kotsokalis, C., Spanoudakis, G., Yahyapour, R.: Establishing and monitoring slas in complex service based systems. In: *IEEE International Conference on Web Services 2009, ICWS 2009*, pp. 783–790, July 2009
5. Dan, A., Davis, D., Kearney, R., Keller, A., King, R., Kuebler, D., Ludwig, H., Polan, M., Spreitzer, M., Youssef, A.: Web services on demand: WSLA-driven automated management. *IBM Syst. J.* **43**, 136–158 (2004)

6. Cloud Management Working Group, D.: Cloud infrastructure management interface (CIMI) model and RESTful HTTP-based protocol an interface for managing cloud infrastructure, dsp0263. Technical report, Distributed Management Task Force, Inc. (2014)
7. Kouki, Y., de Oliveira, F., Dupont, S., Ledoux, T.: A language support for cloud elasticity management. In: 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 206–215, May 2014
8. Liu, F., et al.: SP 500-292 Cloud Computing Reference Architecture, September 2011
9. Ludwig, H., Dan, A., Kearney, R.: Cremona: an architecture and library for creation and monitoring of WS-agreements. In: Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC 2004), pp. 65–74. ACM (2004)
10. Ludwig, H., Keller, A., Dan, A., King, R., Franck, R.: Web Service Level Agreement (WSLA) Language Specification. Technical report, IBM Corporation, January 2003
11. Ludwig, H.: rSLA language specification. Technical report, IBM research Almaden USA, May 2015
12. Mettraux, J.: rufus-scheduler (2005). <https://github.com/jmettraux/rufus-scheduler>. Accessed January 2015
13. Michlmayr, A., Rosenberg, F., Leitner, P., Dustdar, S.: Comprehensive QoS monitoring of web services and event-based SLA violation detection. In: Proceedings of the 4th International Workshop on Middleware for Service Oriented Computing, MWSOC 2009, pp. 1–6. ACM, New York (2009)
14. Muller, C., Oriol, M., Franch, X., Marco, J., Resinas, M., Ruiz-Cortes, A., Rodriguez, M.: Comprehensive explanation of SLA violations at runtime. *IEEE Trans. Serv. Comput.* **7**(2), 168–183 (2014)
15. OCCI: Open Cloud Computing Interface (2010). <http://occi-wg.org/>
16. Torkashvan, M., Haghighi, H.: Cslam: a framework for cloud service level agreement management based on WSLA. In: 2012 Sixth International Symposium on Telecommunications (IST), pp. 577–585, November 2012
17. Tasic, V., Patel, K., Pagurek, B.: WSOL - web service offerings language. In: Bussler, C.J., McIlraith, S.A., Orłowska, M.E., Pernici, B., Yang, J. (eds.) CAiSE 2002 and WES 2002. LNCS, vol. 2512, pp. 57–67. Springer, Heidelberg (2002)
18. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality driven web services composition. In: Proceedings of the 12th International Conference on World Wide Web, WWW 2003, pp. 411–421. ACM, New York (2003)

AISLE: Assessment of Provisioned Service Levels in Public IaaS-Based Database Systems

Jörn Kuhlenkamp^(✉), Kevin Rudolph, and David Bernbach

Information Systems Engineering Research Group,
Technische Universität Berlin, Berlin, Germany
{jk,kr,db}@ise.tu-berlin.de

Abstract. When database systems running on top of public cloud services run into performance problems, it is hard to identify the concrete infrastructure service for which provisioning additional resources would solve said performance problem. In this work, we present AISLE, which develops a model for expected service levels and includes metrics which assess values from service level monitoring to identify these cloud services. Using AISLE, we develop such a model for the Amazon EBS service and evaluate our approach in experiments with Apache Cassandra running on top of EBS-backed EC2 instances.

Keywords: Cloud computing · IaaS · Service levels · Cloud monitoring

1 Introduction

Today's database systems often run on top of IaaS cloud services where the database administrator provisions the required amount of resources for each underlying service. A systematic approach for identifying the required amount is still missing today so that the administrator will typically provision resources based on his gut feeling and adapt later on. Even if the database system is already deployed and running and even if detailed monitoring data is available, it is challenging to identify the specific cloud services where additional resources should be provisioned. This is typically the case since cloud providers offer only very limited Service Level Agreements (SLAs) so that it is hard to interpret monitoring results.

Another problem is that not only the workloads, which the database system is confronted with, change over time requiring adaption of provisioned resource amounts but also the observable service levels of cloud services may vary over time due to a lack of SLAs, e.g., [3].

In this work, we present AISLE (**A**ssessing **I**nfrastructure **S**ervice **L**evel **E**nvironments), an approach that develops a model for an Expected Service Level (ESL) based on SLA information, documentation, and benchmarking results. Part of this approach are three metrics which enable the database administrator to easily assess whether the observed monitoring data for a specific service quality is limited by this expected service level, thus, identifying cloud services

where additional provisioned resources are likely to positively affect the database system. We then use this approach to develop an ESL model for the Amazon EBS service¹ based on documentation and benchmarking. As an evaluation, we deployed an Apache Cassandra cluster on EBS-backed EC2² instances and verified experimentally that provisioning extra resources for the services identified by our metrics yields much better results than other more intuitive scaling approaches.

The paper is structured as follows: Sect. 2 gives a brief introduction to selected cloud services, systems and benchmarking tools. Section 3 describes AISLE, and Sect. 4 presents our application of AISLE to the EBS service. In Sect. 5, we use the model developed in Sect. 4 to evaluate our approach before discussing related work (Sect. 6) and coming to a conclusion (Sect. 7).

2 Background

In this section, we will briefly introduce cloud services, systems, and tools which we will refer to later.

Amazon Web Services: The *Amazon EC2* service offers virtual machines (*EC2 instances*). EC2 instances have an instance type which determines the amount of resources, e.g., virtual CPU cores or memory, available to applications running on that instance. For disk storage, EC2 instances can use either ephemeral disk or EBS volumes, instances of the *Amazon EBS* service. EBS volumes come in different sizes and have a type (e.g., *Provisioned IOPS*) which determines the service level and the amount of dedicated bandwidth.

Apache Cassandra: Cassandra, a NoSQL system initially developed as a write-efficient database for the Facebook message inbox [10], is maintained as Apache project and widely used, e.g., by Netflix. Cassandra implements the Dynamo replication architecture [7] and a BigTable-inspired schemaless data model and storage engine [4].

Flexible I/O Tester: The Flexible I/O Tester (FIO)³ is a tool for studying disk or network performance metrics on Linux systems. FIO works by synthetically creating workloads based on a stochastic model and offers extensive configuration options which makes it particularly suitable for our needs.

Yahoo! Cloud Serving Benchmark: The Yahoo! Cloud Serving Benchmark (YCSB) [5] is an extensible, modular benchmarking tool which emulates clients of a distributed database system and measures client-side performance metrics. YCSB comes with a set of standard workloads of which we use workloads A and B; A is a write-heavy (50% read and 50% update requests), B a read-heavy (95% reads and 5% updates) workload. We use both workloads with a uniform request distribution.

¹ aws.amazon.com/ebs.

² aws.amazon.com/ec2.

³ linux.die.net/man/1/fio.

3 Assessing Cloud Service Levels

In this section, we describe AISLE, which transforms monitoring data into a quality metric-independent score describing how close the specific metric is to an ESL. As cloud providers give very few guarantees in SLAs, we first discuss how a (potentially complex) model – describing service levels a cloud service user could realistically observe – can be developed. Afterwards, we introduce three metrics which describe how a given monitoring value ranks compared to said model. These metrics can then be used to easily identify cloud services where additional resources are likely to have a strong effect on the database system.

3.1 Deriving a Model for Expected Service Levels

Cloud providers rarely offer SLAs that characterize service levels a database administrator can expect from the cloud service(s) the database is running on. If any, these SLAs typically give very limited guarantees [2]. Beyond these *explicitly* guaranteed service levels, additional information can be gained from documentation or via infrastructure benchmarking – this information describes service levels that are not guaranteed but which can still be realistically expected in database deployments. Explicit SLAs will typically be very precise, guaranteeing concrete metric values, e.g., a compute service might guarantee a minimum number of 1.5 Tera Flops per virtual machine. Implicit SLAs, in contrast, could be based on large data sets describing relative frequencies of values measured, e.g., EC2 instances would, based on [12], have a distribution function with two peaks characterizing their computation power. Alternatively, a documentation-based implicit SLA could require a relatively complex model, as we will see later in this paper. In any case, a model for ESLs should describe relative frequencies of service levels which can be expected when using that specific cloud service, i.e., an ESL is only for the most simplistic scenario a single value but will typically be a distribution of values. Also, even the “best”⁴ value of an ESL is not necessarily the “best” value which can be reached – positive outliers are always possible – but it describes service levels that can be realistically reached in practice.

3.2 Normalizing Monitoring Data

For a given Observed Service Level (OSL), which can be obtained through runtime monitoring, we would like to characterize how “close” this value is to the ESL of the corresponding service quality. For quality-independent description and analysis of this “closeness”, we propose to normalize service levels in a way that assigns 100% to observed service levels that reach exactly the ESL. We call this normalized service level *Service Level Pressure* (SLP). Please, note that SLP can reach values beyond 100% whenever an observed service level exceeds the ESL.

⁴ What “best” is, highly depends on the respective quality metric.

The SLP for a given OSL o is defined as the ratio of o to the ESL. If the ESL is a single value then $SLP(o)$ is trivial to calculate (e.g., if ESL is 5 and o is 3, then $SLP(o)$ is 60%). If the ESL is a discrete distribution assigning relative frequencies f to service level values s , then the SLP is calculated as the weighted sum of all ratios o/s , i.e.,

$$SLP(o) = \sum_{\forall s} \frac{o}{s} * f \tag{1}$$

As SLP values are likely to show a large variability over time and short random spikes rarely have an influence on the database system, we propose to introduce the concept of time into two new metrics which are built on top of SLP: For a given time series of SLP values, we are interested in the periods of time during which these SLP values exceed a threshold that the database administrator deems critical, e.g., 70%. Specifically, we propose to use the following two metrics to describe these critical pressure durations:

- *Maximum Critical Pressure Time:* MCPT(c) describes the longest period of time during which the SLP values continuously exceeded the value c .
- *Total Critical Pressure Time:* TCPT(c) describes the total time that the SLP values of the time series exceeded the value c . TCPT(c) is expressed as percentage of the length of the time series.

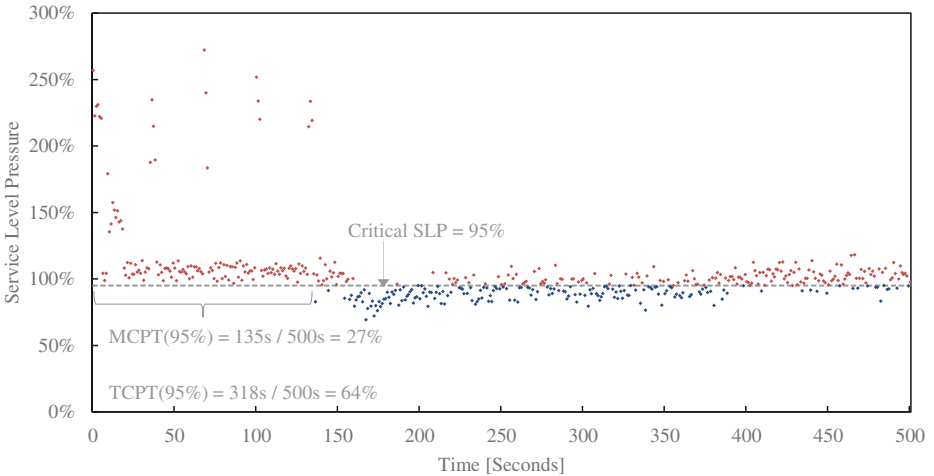


Fig. 1. Maximum critical pressure time and total critical pressure time based on a service level pressure threshold of 95 % for a 500 s interval.

Furthermore, beside using absolute values, we express both metrics as relative durations regarding the overall duration of the analyzed monitoring period in percent. Figure 1 shows an example of MCPT and TCPT. High MCPT values

imply a critical impact, e.g., caused by log consolidation [1] or online data migration [9], on the database system so that the amount of resources provisioned from the corresponding cloud service should be reconsidered. High TCPT values, on the other hand, are caused by the number of requests that the database receives, i.e., they are a hint that indicates from which cloud services the database administrator should provision more resources if the additional load persists.

4 Expected Service Level Model

In this section, we present an ESL model for the Amazon EBS service to apply AISLE to an *infrastructure topology* (topology) that includes EBS volumes. First, we present an analytical ESL model based on the EBS documentation. Second, we present results for an extended experimental evaluation of EBS, our experimental toolkit FIOEBS and an resulting experimental ESL model. Third, we discuss our results and compare analytical and experimental ESL models.

4.1 Analytical Expected Service Level Model

In implicit SLs, Amazon EBS gives throughput guarantees for volumes and connections according to the documentation for the EBS API version 2015-03-01. Precisely, volume throughput is quantified in (i) *normalized operations per second* (no/s) and (ii) bandwidth per second (bw/s). Furthermore, connection throughput is quantified in bw/s. A *normalized operation* (NO) $no(o)$ describes an operation o that is issued against a volume as a function of the operation size in KiB d_o . Therefore, a single operation is divided into a number of NOs based on a *normalized operation size* nd_o . If not stated otherwise, we assume a normalized operation size of $nd_o = 256$ KiB. Therefore, a single operation equals no NOs: $no(o) = \lceil d_o/nd_o \rceil$. For example, an operation o_1 with request size $d_{o_1} = 512$ KiB implies two NO.

An IO volume is provisioned with a configurable number of *provisioned normalized operations* (PNO) per second $pno_{io} = 0, \dots, 20000$. The maximum number of configurable PNO depends on the configured volume size in GiB vs_v of a volume v : $\max pno_{io}(v) = 30 * vs_v$. Furthermore, an IO volume provisions a guaranteed bw/s pbw_{io} : $pbw_{io} = 320$ MiB/s.

EBS-optimized connections provide a dedicated guaranteed bw/s in MiB $pbw_{opt} = 62.5, 93.75, 125, 250, 500$ that depends on the instance type. Non EBS-optimized connections do not provide any explicit performance SLs.

4.2 Experiment Setup

In this section, we present a subset of the benchmarking experiments we used while trying to break and to extend the analytical performance model.

Infrastructure: We conduct all presented experiments in the same EC2 availability zone of *eu-west-1* and use three different EC2 instance types in our

experiments: t2.micro, t2.small, m3.medium. In the tests presented, we used IO Volumes (io1) with sizes of 100 GiB and PNOs of 300, 900 and 1800.

Benchmarking: We use FIO in version 2.1.3 to generate workloads and collect measurements. We deviate from the standard configuration with the following parameters. We emulate synchronous requests, i.e., `ioengine=sync`, disable page cache access, i.e., `direct=1`, refill buffers for operations, i.e., `refill_buffers=1`, and invalidate the buffer-cache, i.e., `invalidate=1`. We generate a synthetic data set of up to 1 file and with a file size of 70 GiB. Before each experiment, we execute a *load phase* in which we create file descriptors and write every block of the complete dataset.

Workload: We use a *Ramp Phase* of 120s in which no measurements are collected to avoid skewed measurements during initialization. Afterwards, we execute a *Run Phase* for 120s during which we collect measurements. We use a single thread to issue operations with an operation size of 4, 32, 64, 256 or 1024 KiB. We issue operations with a random variation of the file offset between issued operations. Furthermore, we use 5 workload mixes with different combinations of read percentage (R) and write percentage (W): (i) R100/W0, (ii) R75/W25, (iii) R50/W50 (iv) R25/W75 and R100/W0.

Measurements: We collect performance measurements with a granularity of 1 s on the user-side, i.e., `Iostat`, and 1 min on the provider-side, i.e., `CloudWatch`. Precisely, we measure throughput, i.e., average number of operations per second (ops/s) and average bandwidth per second (bw/s) in KiB, and, latency, i.e., average per operation latency. Due to space restrictions, latency measurements are not presented in this paper but included in the primary data set.

Implementation of the Experiment Toolkit: We designed and implemented an extensible toolkit called *AisleEbs*⁵ to automate the setup of testbeds and the execution of experiments. Furthermore, we want to foster reproducibility of our experiments. *AisleEbs* allows to setup, execute and aggregate results for a number of FIO experiments on Amazon EC2. We implemented *AisleEbs* based on the configuration management system `Ansible`⁶. Figure 2 provides a concise overview of the interactions of *AisleEbs*.

Experiments: Due to restricted space, we only include a subset of our experiments and datasets. Within this paper, variable parameters in experiments are: instance, instance type, PNOs, workload mix and operation size. The presented results correspond to a total of 1125 experiments that are composed as follows: 1125 experiments = 3 (instance type) × 3 (PNOs) × 5 (operation size) × 5 (workload mix) × 5 (repetitions).

4.3 Experiment Results

Over all experiments, we do not see significant variations for the three parameters instance, instance type and workload mix. Therefore, we present results for

⁵ github.com/jkuhlenkamp/aisle_ebs.

⁶ ansible.com/home.

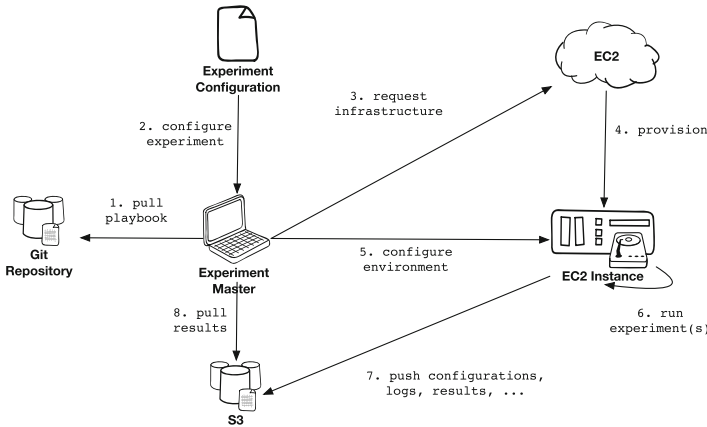


Fig. 2. Overview on interactions within a AisleEbs experiment testbed.

different PNOs and record sizes, i.e., 75 experiments per boxplot. We encourage the interested reader to analyze and/or reproduce our dataset that is available with our toolkit implementation. Presented Boxplots show averages instead of the medians, boxes present 25th and 75th percentils and whiskers 1st and 99th percentils.

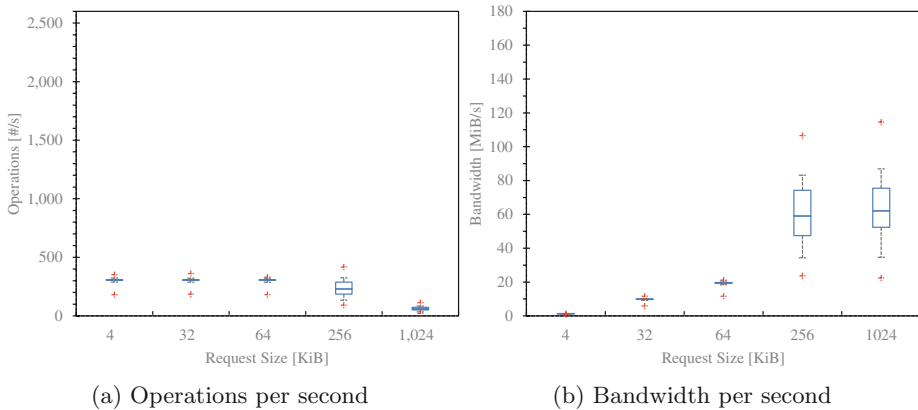


Fig. 3. Throughput for 100 GiB volume size and 300 PNOs.

300 Provisioned Normalized Operations. Figure 3 shows our results for 300 PNOs. Figure 3a shows an avg. throughput of 300 ops/s for operation sizes up to 64 KiB with low variability. With increasing operation sizes avg. ops/s drop down to a minimum of 60 ops/s for 1024 KiB. Figure 3b shows an proportional

increase in avg. bw/s with increasing operation sizes up to 64 KiB with low variability. For operation sizes of 256 KiB and 1024 KiB observed bw/s further increases by a lower factor at large variability.

Our results indicate that up to an operation size of 64 KiB throughput is limited by the number of PNOs. Furthermore, low variability indicates that PNO-based SLs are enforced effectively with low provider-side over- and under-provisioning. For request sizes of 256 KiB and 1024 KiB, our results indicate that throughput is limited by available bw/s. Furthermore, increasing variability indicates that bandwidth-based SLs are not enforced as strictly and/or effectively as operation-based SLs. We argue that the decreasing variability of measured ops/s for operation sizes of 1024 results from the fact that EBS enforces PNOs based on normalized ops/s, and we measure not normalized ops/s as perceived by FIOEBS.

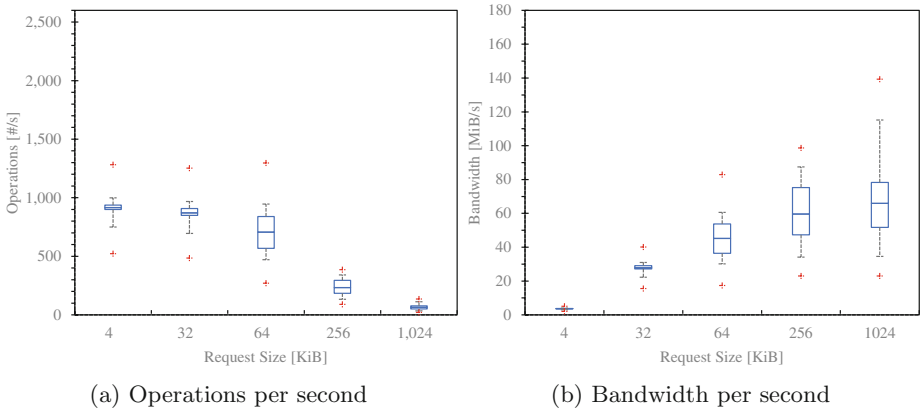


Fig. 4. Throughput for 100 GiB volume size and 900 PNOs.

900 Provisioned Normalized Operations. Figure 4 shows our results for 900 PNOs. Figure 4a shows an avg. throughput of 916 and 871 ops/s with low variability for operations sizes of 4 and 32 KiB. For larger operation sizes, we measure a large decrease of avg. ops/s. Figure 4b shows that proportional bandwidth increase stops and variability increases at operation sizes of 64 KiB. We observe a maximum bandwidth of 139 MiB/s for operations sizes of 1024 KiB. Our results indicate that up to a operation size of 32 KiB throughput is limited by the number of PNOs and for large operation sizes by bandwidth.

1800 Provisioned Normalized Operations. Figure 5 shows our results for 1800 PNOs. Figure 5a shows an avg. throughput of 1788 ops/s for operations sizes of 4 KiB. For larger operation sizes, avg. ops/s drop to 64 for 1024 KiB. Figure 5b shows that proportional bandwidth increase stops and variability increases for operation sizes larger than 4 KiB.

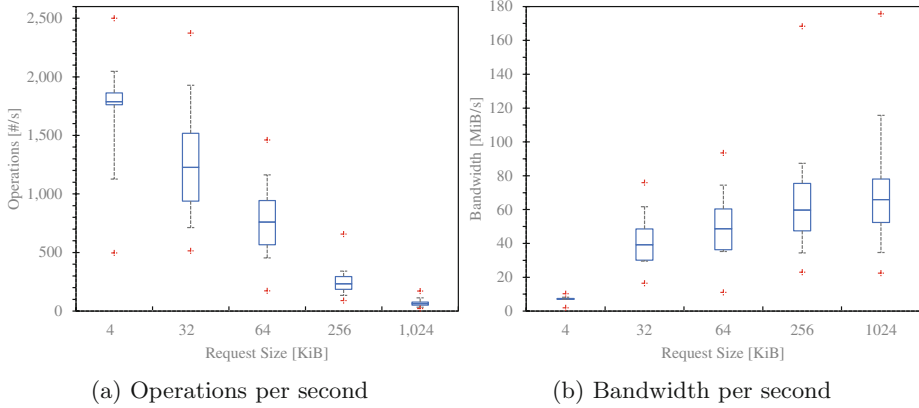


Fig. 5. Throughput for 100 GiB volume size and 1800 PNOs.

Our results indicate that up to a operation size of 4 KiB throughput is limited by the number of PNOs and for large operation sizes by bandwidth. In comparison to 900 PNOs, absolute throughput and throughput variability does not change significantly for operation sizes of 256 KiB and 1024 KiB.

4.4 Discussion

Our results indicate that an increase of PNOs for an volume does not increase avg. bw/s but maximum bw/s for that volume. Furthermore, variability at maximum throughput increases with increasing absolute throughput. We observe the highest throughput variability for the same PNOs at the turning points between operation-based and bandwidth-based throughput limitation. If not subject to an operation-based limitation, we observe an avg. bw/s of 60 MiB for 256 KiB and 66 MiB for 1024 KiB, since indicating that t2.micro, t2.small and m3.medium instances share similar instance to EBS connection limitations. We argue that this observed behavior might change for other instance types and EBS-optimized volume connections, respectively.

For EBS volumes with small numbers of PNOs under workloads with small operation sizes, our results indicate that the analytical ESL model obtained from the EBS documentation predicts ESLs with high accuracy. However, for larger operation sizes and PNOs the accuracy of the analytical model quickly decreases. Overall, we argue that the documentation of ESLs without experimental evaluation is not sufficient to serve as an accurate information base for infrastructure optimization decisions. Experimental evaluation of public infrastructure services increases the overhead for applying AISLE. However, we argue that experimental ESL models can be reused for the evaluation of different deployments and shared between different IaaS users.

Next, we apply AISLE based on the obtained ESL model for EBS in an evaluation and optimization of a Cassandra deployment on EC2.

5 Use Case: Cassandra

In this section, we apply our method and obtained ESL model to a Cassandra deployment on two topologies.

5.1 Deployment Environment

We deployed Cassandra clusters with standard configuration parameters and no replication.

Infrastructure: For each experiment, we provision a 3-node topology of either m3.large or m3.medium EC2 instances, all in the same availability zone of the eu-west-1 region. Both instance types provide moderate networking performance and are not EBS-optimized. We attached a single 100 GiB IO volume with 300 PNOs to each instance.

Workload: As a load generator, we deployed YCSB on a single m3.large instance to emulate 100 client threads running for 30 min. Furthermore, we use for both setups m3.large and m3.medium initial datasets that exceed each instance’s memory almost by a factor three. We emulate YCSB standard workloads A and B.

Toolkit and Measurements: We implemented the tool *AisleCassandra*⁷ to automate experiment execution. We use *Amazon CloudWatch*⁸ to obtain provider-side and *Iostat*⁹ to obtain user-side monitoring measurements.

5.2 AISLE Application

We apply AISLE within our two topology setups under workloads A and B. First, we conduct an a priori analysis of the initial topologies. Second, we adjust the initial topologies and conduct a second a posteriori analysis. For both analyses, we present boxplots of SLP for ops/s (P_{op}) and SLP for bw/s (P_{bw}).

300 Provisioned Normalized Operations. Figure 6 shows our results for setups m3.medium and m3.large for workload A and workload B. For both setups and workloads, our results show a high P_{op} compared to a low P_{bw} . m3.large under both workloads and m3.medium under workload B show a P_{op} above 80% on avg. and 92% for the 75th percentile for all volumes except volume 1 in m3.medium. Since, indicating underprovisioned SLs regarding ops/s SLs. m3.medium under workload A shows 99th percentile P_{op} of less than 85% for volume 1 and 3, since, results indicate small potential for optimization for both volumes. We adjust the volumes in both setups regarding ops/s SLs. Precisely, we increase PNOs from 300 to 1800 for all IO volumes.

⁷ github.com/jkuhlenkamp/aisle_cassandra.

⁸ aws.amazon.com/cloudwatch/.

⁹ linux.die.net/man/1/iostat.

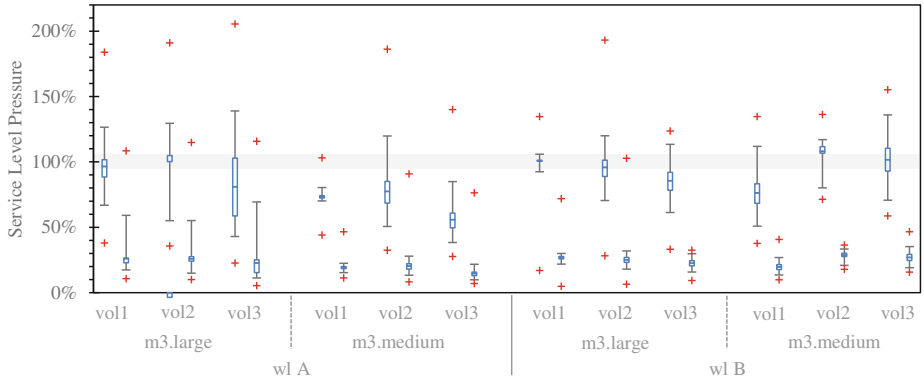


Fig. 6. Boxplots of Service Level Pressures for (i) ops/s and (ii) bw/s for EBS volumes in 3 node clusters and 300 Provisioned Normalized Operations.

1800 Provisioned Normalized Operations. Figure 7 shows our results for the adjusted setups m3.medium and m3.large for workload A and workload B. For both workloads and adjusted setups, our results show a shift from a high P_{op} to high P_{bw} in comparison to the initial setups. For both workloads, all volumes in m3.large report 25th percentile P_{bw} above 89%. Therefore, our results indicate underprovisioned SLs regarding bw/s. For both workloads, all volumes in m3.medium report 99th percentile values for P_{op} and P_{bw} of less than 71%. Therefore, indicating overprovisioned volumes regarding both SLs.

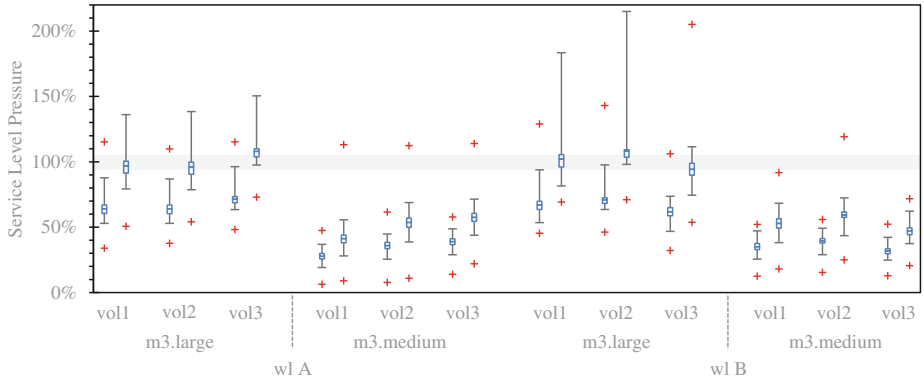


Fig. 7. Boxplots of Service Level Pressures for (i) ops/s and (ii) bw/s for EBS volumes in 3 node clusters and 1800 Provisioned Normalized Operations.

5.3 Discussion

We showed that AISLE is suitable to characterize the optimization potential regarding SLs of public IaaS-based topologies. However, we are aware that our approach might imply a certain implementation and management overhead in comparison to a trial and error-based scale-out strategy. The overhead is due to obtaining ESL models, the setup of monitoring infrastructure and conducting analysis. Therefore, we compare AISLE to a scale-out strategy. Furthermore, we discuss the adoption of provider-side monitoring solutions to decrease required management overheads for monitoring solutions.

AISLE vs. Scale-Out. The application of AISLE implies an additional management overhead for a user of an infrastructure service. Therefore, we compare AISLE to a trivial scale-out strategy, i.e., the provisioning of additional servers with the same configuration into an existing cluster. We refer to AISLE-based scaling as *Option A* (Experimental) and scale-out-based scaling as *Option B* (Analytical). We compare both approaches in terms of application SLs for the Cassandra cluster using a ratio for simpler comparison. Precisely, the ratio consists of the overall topology cost in \$ per month divided by Cassandra throughput, i.e. average client requests per second (req/s). For Option B, we assume a linear increase in throughput as indicated by previous research [9, 14]. Furthermore, we select the corresponding cluster size for Option B as follows: We select the highest resulting throughput that is lower than the corresponding throughput for Option A. Table 1 shows a concise comparison between Option A and Option B. As shown in column *Opt. A - Opt. B* and indicated by negative comparison values, a AISLE-based scaling strategy results in a higher scaleup under lower costs per month for all workloads (wl A and wl B) and instances types and, as expected, a higher benefit for m3.large.

Table 1. Performance and costs comparison for scaling an initial Cassandra cluster (Base) with scaling strategy AISLE (Opt.A) vs. horizontal scale-out (Opt.B).

	Opt. A (Experimental)		Opt. B (Analytical)		Opt. A - Opt. B
	PIOPS = 1,800		PIOPS = 300		Savings
	#nodes	\$/ (req/s)	#nodes	\$/ (req/s)	\$/ (req/s)
m3.large, wl A (update-heavy)	3	0.08	14	0.22	-0.14
m3.large, wl B (read-heavy)	3	0.13	13	0.33	-0.20
m3.medium, wl A (update-heavy)	3	0.11	7	0.13	-0.02
m3.medium, wl B (read-heavy)	3	0.19	7	0.23	-0.04

Provider-Side vs. Client-Side Monitoring. AISLE requires monitoring of infrastructure resources. Therefore, we compare two common options that are available to a user to enable a monitoring solution: (i) client-side monitoring, i.e., Iostat, and (ii) provider-side monitoring, i.e., Amazon CloudWatch. Client-side monitoring implies a management and performance overhead. Still, client-side monitoring implies the freedom to adapt and verify monitored metrics and granularity. Provider-side monitoring implies a low management and performance overhead. However, users have limited control over the monitoring solution and must trust a provider.

Next, we apply AISLE with monitoring traces obtained from Amazon CloudWatch. According to the latest CloudWatch documentation, i.e., API version 2010-08-01, CloudWatch provides a metric `VolumeConsumedReadWriteOps` that reports no/s based on a normalized operation size of 256 KiB with 1 min granularity. Figure 8 shows no/s reported by CloudWatch and Iostat for a single volume. Our measurements indicate that CloudWatch reports much higher no/s for the same 1 min interval. We further analyzed this behavior. We believe that CloudWatch reports no/s based on a normalized operation size of 16 KiB instead of 256 KiB. Therefore, we derived no/s for normalized operation size of 16 KiB for Iostat measurements. Figure 8 shows the derived scaling factor and the resulting no/s model for a normalized operation size of 16 KiB for Iostat. Visual analysis indicates that this is indeed the case.

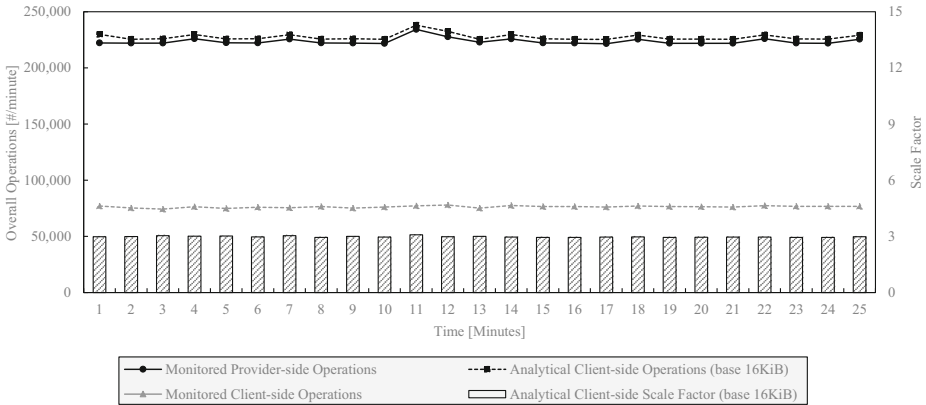


Fig. 8. Comparison of client-side (Iostat) and provider-side (CloudWatch) throughput measurements for different normalized operation sizes.

6 Related Work

There is no directly related approach with the same scope as AISLE. Existing publications fall in either of the following three groups:

Infrastructure Optimization: Several autonomous computing approaches adapt provisioned resources at runtime for database systems, e.g., [6, 13, 17]. In contrast to AISLE, these approaches all require explicit knowledge on the implementation details of the database systems whereas AISLE is theoretically application-agnostic – even though only evaluated for database systems – and treats the system as a blackbox. As AISLE in its current version does not adapt provisioned resources but rather provides the necessary information to do so to an administrator, these approaches complement our work.

Provider Perspective: Approaches like [16, 18] take a provider perspective and focus, e.g., on offering optimal physical resource sets to virtual machines depending on the runtime requirements of the virtual machine. AISLE, in contrast, takes the perspective of a client who does not have any influence and knowledge on the service internals.

Infrastructure Benchmarking: Performance benchmarking of infrastructure services like Amazon EBS has been done before, e.g., [8, 11, 15]. We introduce new metrics to not only measure but also interpret results. For the special case of Amazon EBS, we also ran our experiments in more configuration setups and could, thus, also observe more fine-grained results: For instance, our experiments indicate low bandwidth variability for a certain combinations of EBS volume configurations and operation sizes whereas [11] reported overall high bandwidth variability.

7 Conclusion

In this work, we presented AISLE, an approach that develops a model for an Expected Service Level based on SLA information, documentation, and benchmarking results. Part of this approach are three metrics which enable the database administrator to easily assess whether the observed monitoring data for a specific service quality is limited by this expected service level, thus, identifying cloud services where additional provisioned resources are likely to positively affect the database system. As an example, we have based on AISLE developed an ESL node for the Amazon EBS service and verified experimentally that provisioning extra resources for the services identified by our metrics yields much better results than other more intuitive scaling approaches. We believe that our approach is not limited to database systems and will, therefore, in future work try to extend it to all kinds of applications running on top of cloud services.

Acknowledgments. We thank Amazon Web Services for a generous grant that enabled us to run our experiments.

References

1. Ahmad, M.Y., Kemme, B.: Compaction management in distributed key-value datastores. *PVLDB* **8**(8), 850–861 (2015)

2. Baset, S.A.: Cloud SLAs. *ACM SIGOPS Operating Syst. Rev.* **46**(2), 57 (2012)
3. Bermbach, D., Tai, S.: Benchmarking eventual consistency: Lessons learned from long-term experimental studies. In: *Proceedings of the 2nd International Conference on Cloud Engineering (IC2E)*. IEEE (2014)
4. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: a distributed storage system for structured data. *ACM Trans. Comput. Syst.* **26**(2), 1–26 (2008)
5. Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.: Benchmarking cloud serving systems with YCSB. In: *Proceedings of the 1st ACM Symposium on Cloud Computing*
6. Copil, G., Trihinas, D., Truong, H.-L., Moldovan, D., Pallis, G., Dustdar, S., Dikaiakos, M.: ADVISE – a framework for evaluating cloud service elasticity behavior. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) *ICSOC 2014. LNCS*, vol. 8831, pp. 275–290. Springer, Heidelberg (2014)
7. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Voshall, P., Vogels, W.: Dynamo: amazon’s highly available key-value store. In: *Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles*
8. Dittrich, J., Quian, J.A.: Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proc. VLDB Endow.* **3**(1), 460–471 (2010)
9. Kuhlenkamp, J., Klems, M., Röss, O.: Benchmarking scalability and elasticity of distributed database systems. *Proc. VLDB Endow.* **7**(12), 1219–1230 (2014)
10. Lakshman, A., Malik, P.: Cassandra. *ACM SIGOPS Operating Syst. Rev.* **44**(2), 35–40 (2010)
11. Leitner, P., Cito, J.: Patterns in the chaos a study of performance variation and predictability in public IaaS clouds. In: *Proceedings of the 24th International World Wide Web Conference (WWW 2015)*. ACM, Florence (2015)
12. Lenk, A., Menzel, M., Lipsky, J., Tai, S., Offermann, P.: What are you paying for? performance benchmarking for infrastructure-as-a-service offerings. In: *2011 IEEE International Conference on Cloud Computing (CLOUD)*, pp. 484–491 (2011)
13. Lim, H.C., Babu, S., Chase, J., Parekh, S.: Automated control in cloud computing: challenges and opportunities. In: *Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds*, pp. 13–18 (2009)
14. Rabl, T., Sadoghi, M., Jacobsen, H.A., Gómez-Villamor, S., Muntés-Mulero, V., Mankowskii, S.: Solving big data challenges for enterprise application performance management. *Proc. VLDB Endow.* **5**(12), 1724–1735 (2012)
15. Scheuner, J., Leitner, P., Cito, J., Gall, H.: Cloud WorkBench - infrastructure-as-code based cloud benchmarking. In: *Proceedings of the 6th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2014)* (2014)
16. Shen, Z., Subbiah, S., Gu, X., Wilkes, J.: CloudScale. In: *Proceedings of the 2nd ACM Symposium on Cloud Computing - SOCC 2011*, pp. 1–14 (2011)
17. Trushkowsky, B., Fox, A., Franklin, M.: The scads director: Scaling a distributed storage system under stringent performance requirements. In: *Proceedings of the 9th USENIX Conference on File and Storage Technologies (FAST 2011)*, pp. 12–25. USENIX Association, Berkeley (2011)
18. Yao, J., Jung, G.: Bottleneck detection and solution recommendation for cloud-based multi-tier application. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) *ICSOC 2014. LNCS*, vol. 8831, pp. 470–477. Springer, Heidelberg (2014)

Service Composition

Are RESTful APIs Well-Designed? Detection of their Linguistic (Anti)Patterns

Francis Palma^{1,2}(✉), Javier Gonzalez-Huerta¹, Naouel Moha¹,
Yann-Gaël Guéhéneuc², and Guy Tremblay¹

¹ Département d'informatique, Université du Québec à Montréal, Montréal, Canada
{gonzalez.huerta.javier,moha.naouel,tremblay.guy}@uqam.ca

² Ptidej Team, DGIGL, École Polytechnique de Montréal, Montréal, Canada
{francis.palma,yann-gael.gueheneuc}@polymtl.ca

Abstract. Identifier lexicon has a direct impact on software understandability and reusability and, thus, on the quality of the final software product. Understandability and reusability are two important characteristics of software quality. REST (REpresentational State Transfer) style is becoming a *de facto* standard adopted by many software organisations. The use of proper lexicon in RESTful APIs might make them easier to understand and reuse by client developers, and thus, would ease their adoption. *Linguistic antipatterns* represent poor practices in the naming, documentation, and choice of identifiers in the APIs as opposed to *linguistic patterns* that represent best practices. We present the DOLAR approach (Detection Of Linguistic Antipatterns in REST), which applies syntactic and semantic analyses for the detection of linguistic (anti)patterns in RESTful APIs. We provide detailed definitions of ten (anti)patterns and define and apply their detection algorithms on 15 widely-used RESTful APIs, including Facebook, Twitter, and YouTube. The results show that DOLAR can indeed detect linguistic (anti)patterns with high accuracy and that they do occur in major RESTful APIs.

Keywords: REST · Patterns · Antipatterns · Detection · Semantic analysis

1 Introduction

Service-Oriented Architecture (SOA) has changed the way software systems are developed, deployed, and consumed [6]. The REpresentational State Transfer (REST) architectural-style [7] is becoming a *de facto* standard, adopted by large software organisations like Facebook, Twitter, Dropbox, and YouTube, for developing and publishing their services, *a.k.a.* their RESTful APIs.

In REST, well-designed URIs (Uniform Resource Identifiers) facilitate maintenance and evolution for APIs developers. Moreover, well-designed and named RESTful APIs may attract client developers more than poorly designed or named ones [14] because client developers must understand the providers' APIs while

designing and developing their Web-based systems that use these APIs. Therefore, in the design and development of RESTful APIs, their understandability and reusability are two major quality factors.

Source code lexicon is shown to be an influential factor on the understandability, reusability and, overall, on the quality of software systems [12]. APIs designers use related natural names—natural language words—to name software entities [11]. In REST, linguistic relations among resources, services, and parameters are crucial [8] and the lack of such linguistic relations and/or poor naming may degrade the overall design of RESTful APIs and translate into *linguistic antipatterns*. Linguistic antipatterns are *poor* solutions to common recurring naming problems, which may hinder the consumption of RESTful APIs. In contrast, *linguistic patterns* are *best* solutions to common naming problems and may facilitate the consumption of RESTful APIs.

A number of *best* and *poor* linguistic practices for RESTful APIs design are listed in the literature [4,8,14] but they do not provide clear and detailed descriptions. In this paper, we represent those best and poor practices as patterns and antipatterns, respectively. For example, ✘ *Contextless Resource Names* [8] is a linguistic antipattern that describes a URI composed of nodes from different semantic contexts as in the URI www.example.com/newspaper/player where “newspaper” and “player” do not belong to the same semantic context. On the contrary, ✔ *Contextualised Resource Names* [8] is a linguistic pattern describing a URI composed of nodes that belong to the same semantic context and helping developers to understand better the resources or the interaction context with the server, and thus, increasing the understandability and reusability of an API. An example URI is www.example.com/newspapers/media because “newspapers” and “media” belong to the same semantic context. For RESTful APIs, the automatic detection of such linguistic patterns and antipatterns is a means to assess their understandability and reusability. However, no previous work analysed linguistic (anti)patterns in RESTful APIs.

In this paper, we present DOLAR (Detection Of Linguistic Antipatterns in REST), an approach supported by SOFA (Service Oriented Framework for Antipatterns) [17], which integrates syntactic and semantic analyses of RESTful APIs for detecting linguistic (anti)patterns. Semantic analyses are used to infer meaning and relationships among language elements whereas syntactic analyses focus on structural properties [9]. We propose (1) a detailed definition of ten common (anti)patterns for RESTful APIs [4,8,14] and their corresponding detection algorithms; (2) the DOLAR approach relying on the SOFA framework [17] extended with syntactic and semantic analyses based on WordNet¹ and Stanford CoreNLP²; (3) an empirical validation of DOLAR in which we analyse ten REST linguistic (anti)patterns on a set of 15 well-known RESTful APIs—including Facebook, Twitter, and YouTube—invoking over 300 methods. The validation results show that (1) DOLAR has an average precision and recall over 75% and (2) out of the 15 analysed RESTful APIs, most of them involve

¹ wordnet.princeton.edu.

² nlp.stanford.edu/software/corenlp.shtml.

syntactical URIs design problems and they do not organise URIs nodes in a hierarchical manner. Moreover, we also observed that the REST APIs designers, in general, use appropriate contextual resource names and they do not use verbs in URIs, which is a good URIs design practice in REST.

The remainder of the paper is organised as follows: Sect. 2 discusses related work. Section 3 presents the ten linguistic (anti)patterns. Section 4 presents the DOLAR approach. Section 5 presents a validation of DOLAR. Finally, Sect. 6 concludes the paper and sketches future work.

2 Related Work

Over the last years, several researchers (*e.g.*, [1,2]) used semantic analyses to detect linguistic antipatterns and to check for consistency between source code and comments in object-oriented (OO) systems.

Abebe *et al.* [1] present a first set of lexicon bad smells in OO source code and a tool-suite that uses semantic analyses for their detection. Arnaoudova *et al.* [2] present a first definition of *linguistic antipatterns*, define 17 linguistic antipatterns in OO programming, and implement their detection algorithms. The authors search for the differences between the naming used for software entities (*e.g.*, method names and return types) and their implementation and/or documentation. For example, one antipattern they define “*Is*” returns more than a *Boolean*, which analyses the name of a method starting with “*Is*” and checks whether the method returns a boolean or not [2].

Semantic analyses are also applied to Web services design and development [15, 21]. Rodriguez *et al.* [21] present a study on bad linguistic practices identified on a set of WSDL descriptions and provide a catalog of Web services discoverability antipatterns. These antipatterns focus on the comments, elements names, or types used for representing the data models in WSDL documents. Mateos *et al.* [15] present a tool to detect a subset of antipatterns presented in [21].

Other researchers also use semantic analyses in different aspects of the software development life-cycle [3,13,20]. For example, Lu *et al.* [13] define an approach to improve code searches by identifying relevant synonyms using the WordNet English lexical database. Arnaoudova *et al.* [3] perform analyses on identifiers renaming in OO systems and classify them. Finally, Rahman and Roy [20] present an approach to automatically suggest relevant search terms based on the textual descriptions of software change tasks.

These approaches are tailored to OO identifiers and their consistencies with comments [1,2] or to traditional SOAP-based Web services interfaces [15,21], and therefore, they cannot be applied to RESTful APIs due to their intrinsic nature. For example, the invocation of RESTful services relies on a uniform interface formed using HTTP methods to access or modify resources via URIs.

Some researchers have dealt with the linguistic aspect of RESTful APIs. For example, Hausenblas [10] performs a subjective analysis on RESTful APIs to assess the quality of the URIs naming. However, he does not perform an automatic nor a systematic analysis. Moreover, he does not search for specific

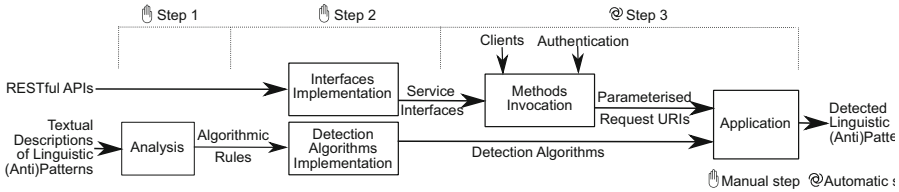


Fig. 1. DOLAR approach.

antipatterns. Parrish [19] also performs a subjective lexical comparison between two well-known RESTful APIs, *e.g.*, Facebook and Twitter. In the comparison, the author analyses, for example, the use of verbs and nouns in URIs naming.

Although the above two deal with linguistic aspects of RESTful APIs, they only rely on the subjective view on a set of good linguistic practices and recommendations. Thus, there is no dedicated approach to automatically assess the linguistic quality of RESTful APIs by detecting poor and best practices.

3 REST Linguistic Patterns and Antipatterns

Table 1 presents the ten linguistic (anti)patterns that we consider in this paper and that have been extracted from existing literature [4, 8, 14, 23].

4 The DOLAR Approach

We now present the DOLAR approach (Detection Of Linguistic Antipatterns in REST) for the analysis and detection of linguistic (anti)patterns in RESTful APIs. DOLAR proceeds in three steps, as shown in Fig. 1.

Step 1. Analysis of Linguistic (Anti)Patterns: This step consists in analysing the description of REST linguistic (anti)patterns from the literature to identify their relevant properties. We use these relevant properties to define *algorithmic rules* for (anti)patterns.

Step 2. Implementation of Interfaces and Detection Algorithms: This step involves the implementation of detection algorithms for (anti)patterns based on rules defined in Step 1 and the service interfaces for RESTful APIs.

Step 3. Detection of Linguistic (Anti)Patterns: This step deals with the automatic application of detection algorithms implemented in Step 2 on RESTful APIs for the detection of linguistic (anti)patterns.

4.1 Analysis of Linguistic Patterns and Antipatterns

We analyse the definitions of the (anti)patterns listed in Sect. 3 to identify their linguistic aspects. A linguistic aspect for the detection of the *Context-less Resource Names* antipattern is, for example, to check if a URI nodes belong to the same semantic context.

Table 1. List of ten linguistic (anti)patterns in REST.

1. ✓Contextualised vs. ✗Contextless Resource Names
Description: URIs should be <i>contextual</i> , <i>i.e.</i> , nodes in URIs should belong to semantically-related context. Thus, the <i>Contextless Resource Names</i> antipattern appears when URIs are composed of nodes that do not belong to the same semantic context.
Example: ✗ https://www.example.com/newspapers/players?id=123 is a ✗ <i>Contextless Resource Names</i> antipattern because ‘newspapers’ and ‘players’ do not belong to same semantic context. ✓ https://www.example.com/newspapers/media/page?id=123 is a ✓ <i>Contextual Resource Names</i> pattern because ‘soccer’, ‘team’, and ‘players’ belong to same semantic context.
Consequences: <i>Contextless Resource Names</i> do not provide a clear context for a request, which may mislead the APIs clients by decreasing the understandability of the APIs [8].
2. ✓Hierarchical vs. ✗Non-hierarchical Nodes
Description: Each node forming a URI should be hierarchically related to its neighbor nodes. In contrast, <i>Non-hierarchical Nodes</i> is an antipattern that appears when at least one node in a URI is not hierarchically related to its neighbor nodes.
Example: ✗ https://www.example.com/professors/university/faculty/ is a ✗ <i>Non-hierarchical Nodes</i> antipattern since ‘professors’, ‘faculty’, and ‘university’ are not in a hierarchical relationship. ✓ https://www.example.com/university/faculty/professors/ is a ✓ <i>Hierarchical Nodes</i> pattern since ‘university’, ‘faculty’, and ‘professors’ are in a hierarchical relationship.
Consequences: Using non-hierarchical names may confuse users on the real purpose of the API and hinders their understandability and, therefore, the API’s usability [8].
3. ✓Tidy vs. ✗Amorphous URIs
Description: REST resource URIs should be tidy and easy to read. A <i>Tidy URI</i> is a URI with appropriate lower-case resource naming, no extensions, underscores, or trailing slashes. <i>Amorphous URI</i> antipatterns appear when URIs contain symbols or capital letters that make them difficult to read and use. As opposed to good practices [14], a URI is amorphous if it contains: (1) upper-case letter (except for Camel Cases [16]), (2) file extensions, (3) underscores, and, (4) a final trailing-slash.
Example: ✗ https://www.example.com/NEW.Customer/_photo01.jpg/ is a ✗ <i>Amorphous URI</i> antipattern since it includes a file extension, upper-case resource names, and underscores. ✓ https://www.example.com/customers/1234 is a ✓ <i>Tidy URI</i> pattern since it only contains lower-case resource naming, without extensions, underscores, or trailing slashes.
Consequences: (1) Upper/lower-case names may refer to different resources, RFC 3986 [4]. (2) File extensions in URIs violate RFC 3986 and affect service evolution. (3) Underscores are hidden when highlighting URIs, decreasing readability. (4) Trailing-slash mislead users to provide more resources.
4. ✓Verbless vs. ✗CRUDy URIs
Description: Appropriate HTTP methods, <i>e.g.</i> , GET, POST, PUT, or DELETE, should be used in <i>Verbless URIs</i> instead of using CRUDy terms (<i>e.g.</i> , create, read, update, delete, or their synonyms) [8]. The use of such terms as resource names or requested actions is highly discouraged [14].
Example: ✗POST https://www.example.com/update/players/age?id=123 is a ✗ <i>CRUDy URIs</i> antipattern since it contains a CRUDy term ‘update’ while updating the user’s profile color relying on an HTTP POST method. ✓POST https://www.example.com/players/age?id=123 is a ✓ <i>Verbless URIs</i> pattern since is an HTTP POST request without any verb.
Consequences: Using CRUDy terms in URIs can be confusing for API clients, <i>i.e.</i> , in the best cases they overload the HTTP methods and in the worst cases they go against HTTP methods. CRUDy terms in a URI confuse and prohibit users to use proper HTTP methods in a certain context and may introduce another REST antipattern, <i>Tunnelling through GET/POST</i> [23].
5. ✓Singularised vs. ✗Pluralised Nodes
Description: URIs should use singular/plural nouns consistently for resources naming across the API. When clients send PUT/DELETE requests, the last node of the request URI should be singular. In contrast, for POST requests, the last node should be plural. Therefore, the <i>Pluralised Nodes</i> antipattern appears when plural names are used for PUT/DELETE requests or singular names are used for POST requests. However, GET requests are not affected by this antipattern [8].
Example: The first example URI is a POST method that does not use a pluralised resource, thus leading to ✗ <i>Pluralised Nodes</i> antipattern. On the other hand, for the ✓ <i>Singularised Nodes</i> pattern, the DELETE request acts on a single resource for deleting it. ✗DELETE https://www.example.com/team/players or ✗POST https://www.example.com/team/player ✓DELETE https://www.example.com/team/players or ✓POST https://www.example.com/team/players
Consequences: If a plural node for PUT (or DELETE) request is used at the end of a URI, the API clients cannot create (or delete) a collection of resources, which may result in, for example, a 403 Forbidden server response. In addition, even if the resources can be filtered through query-like parameters, it confuse the user if one or multiple resources are being accessed/deleted [8].

```

1: CONTEXTLESS-RESOURCE-NAMES(Request-URI)
2:  URINodes ← EXTRACT-URI-NODES(Request-URI)
3:  for each index = 1 to LENGTH(URINodes)-1
4:    Set1 ← CAPTURE-CONTEXT-BY-SYNSETS(URINodesindex)
5:    Set2 ← CAPTURE-CONTEXT-BY-SYNSETS(URINodesindex+1)
6:    if Set1 ∩ Set2 = ∅
7:      print “Contextless Resource Names detected”
8:      break
9:    end if
10:  end for

```

Fig. 2. Algorithmic rule of the *Contextless Resource Names* antipattern.

Figure 2 shows the algorithmic rule we define for the *Contextless Resource Names* antipattern. We compare the context of every pair of nodes or resources in a URI, lines 4–6. We report a URI as an occurrence of this antipattern if we find at least one contextless relation among all possible resource pairs. Conversely, we report an occurrence of the corresponding pattern *iff* all possible resource pairs share at least one common context and are relevant for that particular URI.

We rely on WordNet and Stanford CoreNLP to capture contexts and perform semantic analyses. WordNet is a widely used lexical database, which groups nouns, verbs, and adjectives into sets of cognitive synonyms—*synsets*—each representing unique concepts which can be used interchangeably in a certain context. WordNet is useful in finding semantic similarity between words using its underlying *hypernym-hyponym* and *meronym-holonym* relations as Fig. 3 depicts. In Fig. 3a, *medium* is one of 11 synsets of ‘media’ and there exist different types of *medium* including *newspaper*, *film*, *telecommunication*, and so on defined in WordNet. Based on WordNet, *medium* is thus the *hypernym* of *newspaper* and *newspaper* is the *hyponym* of *medium*. Such relations also exhibit contextual relevance between words and can be useful for analysing *Contextless Resource Names* antipattern [8] in URIs. In addition, there exist *part-of*, *i.e.*, *holonym-meronym*, relations between words defined in WordNet (see Fig. 3b). For example, a *university* consists of *faculty member*, *student*, and *department* and the *department* may include *biology* and *chemistry*. Thus, *university* is a *holonym* of *faculty member* and *faculty member* is a *meronym* of *university*. Such hierarchical relations defined in WordNet between words can be useful in analysing *Non-hierarchical Nodes* antipattern [8].

Moreover, Stanford’s CoreNLP annotate nodes (after splitting CamelCase nodes) with its underlying POS (part-of-speech) tagger to differentiate verbs (*i.e.*, actions) and nouns (*i.e.*, resources). We also define algorithmic rules for nine other linguistic (anti)patterns.

4.2 Implementation of Interfaces and Detection Algorithms

This step includes the implementation of services’ interfaces and the implementation of detection algorithms of linguistic (anti)patterns. We implemented the service interfaces of RESTful APIs under study using JAVA, which contain the

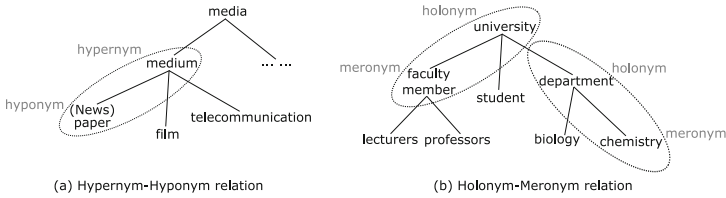


Fig. 3. *Hypernym-Hyponym and Meronym-Holonym relations in WordNet.*

methods callable to access or modify services' underlying resources. Each of the interface methods is mapped to a HTTP method. Using the appropriate HTTP methods, our DOLAR approach sends HTTP requests to real RESTful APIs and receives HTTP responses. Linguistic (anti)patterns, for example, *Amorphous URIs* (or *Tidy URIs*) require the fully-parameterised request URIs to be detected, which can only be obtained after HTTP requests are made. For each RESTful API, the details required to implement its service interfaces, *i.e.*, resources, HTTP actions to perform on its resources, and the parameters for each HTTP request, can be found in its online documentation as shown in Table 2. For other linguistic (anti)patterns, it is enough to extract URIs from the documentation of the RESTful APIs and then to analyse them.

Like the REST service interfaces, the detection algorithms for linguistic (anti)patterns are also written in JAVA. In fact, we manually transform the algorithmic rules defined the previous section into the executable programs.

4.3 Detection of Linguistic Patterns and Antipatterns

Methods Invocation: For each RESTful API, besides the service interfaces, we also implement clients to call the methods in the service interfaces, which perform read, write, update, or delete operations on resources. These explicit calls are done at detection time to obtain fully parameterised request URIs sent to the servers, which are required for detecting (anti)patterns like *Amorphous URI*. In REST, a resource may be related to multiple Java methods because any of the four basic operations (GET, POST, PUT, and DELETE) can be performed. As for the clients authentication, large companies often requires clients authentication to accept secured HTTP requests. Thus, we also implement the *OAuth 2.0* authentication protocol. In the end, this step produces the set of all parameterised requests URIs and their responses.

Application of Detection Algorithms: The SOFA framework (Service Oriented Framework for Antipatterns) [17] automatically applies the algorithmic rules in the form of detection algorithms on the parameterised requests URIs from the clients, collected in the previous step. Finally, the SOFA framework returns a set of detected REST linguistic (anti)patterns.

The SOFA framework, uses a Service Component Architecture (SCA) [5]. It relies on FraSCAti [22] for its runtime support. We added 13 REST (anti)patterns

related to the design of REST requests/responses in a previous work [18]. We extend SOFA with detection support of REST linguistic (anti)patterns using linguistic analyses based on WordNet and Stanford CoreNLP.

Specifically, we extend the *REST Handler* component to facilitate the detection of REST linguistic (anti)patterns by wrapping each RESTful API in an SCA component and applying the detection algorithms on the SCA-wrapped RESTful APIs. By wrapping each API, we can introspect each full request URI with its actual runtime parameters, relying on *FraSCAti IntentHandler*, a runtime interceptor. We invoke methods from a service interface defined with an *IntentHandler* to introspect the request details, which allows on-the-fly syntactic and semantic analyses of parameterised request URIs.

5 Validation

In this section, we assess the effectiveness of DOLAR approach by showing the accuracy of the defined algorithmic rules, the extensibility of our SOFA framework, and the performance of the detection algorithms.

5.1 Hypotheses

We define three hypotheses to assess DOLAR’s effectiveness:

H₁. Accuracy: *The set of all defined rules have an average precision and recall of more than 75 %, i.e., more than three out of four are true positives and we do not miss more than one out of four of all existing (anti)patterns.*

H₂. Extensibility: *Our SOFA framework is extensible for adding new service-oriented and REST-specific (anti)patterns. In addition, SOFA facilitates an easy integration of new RESTful APIs.*

H₃. Performance: *The concretely implemented detection algorithms perform with a low detection times, i.e., on an average in the order of seconds.*

5.2 Subjects and Objects

The subjects of our study are the ten REST linguistic (anti)patterns described in Sect. 3. The objects are 15 common and well-known RESTful APIs for which we found documentations. We choose APIs whose underlying HTTP methods, APIs end-points, and authentication mechanisms are well explained, for example Facebook, Twitter, Dropbox, or YouTube, as summarised in Table 2.

5.3 Validation Process

We followed the instructions in the online documentation for APIs and implemented their (authenticated) clients. We invoked a set of 309 REST methods from 15 RESTful APIs to access their resources. We collected all fully parameterised request URIs from the clients and responses from the servers. Later, we

Table 2. List of 15 analysed RESTful APIs and their online documentations.

RESTful APIs	Online documentations
Alchemy	alchemyapi.com/api
BestBuy	developer.bestbuy.com/documentation
Bitly	dev.bitly.com/api.html
CharlieHarvey	charlieharvey.org.uk/about/api
Dropbox	dropbox.com/developers/core/docs
Externalip	api.externalip.net
Facebook	developers.facebook.com/docs/graph-api
Instagram	instagram.com/developer
Musicgraph	developer.musicgraph.com/api-docs/overview
Ohloh	github.com/blackducksw/ohloh_api
StackExchange	api.stackexchange.com/docs
TeamViewer	integrate.teamviewer.com/en/develop/documentation
Twitter	dev.twitter.com/rest/public
YouTube	youtube.com/yt/dev/api-resources.html
Zappos	developer.zappos.com/docs/api-documentation

applied our algorithmic rules in the form of detection algorithms implemented manually on the REST requests URIs and report (anti)patterns detected by our SOFA framework. We validated the results in two phases: (1) all the Dropbox URIs and (2) four representative APIs, *i.e.*, Facebook, Twitter, Dropbox, and YouTube, for which we randomly selected some candidate request URIs detected as (anti)patterns. We chose those four APIs based on our previous findings [18], which concluded that Twitter and Dropbox are more problematic APIs, whereas Facebook and YouTube were well-designed.

We involved three professionals manually evaluated the URIs to identify the true positives and false negatives to define a ground truth for a predefined subset of the analysed URIs. The professionals have knowledge on REST and did not take part in the detection step. We provided them with the descriptions of REST linguistics (anti)patterns and the sets of all requests URIs collected during the service invocations. We resolved conflicts at the majority.

Due to the large size of the data-sets, we performed the validation on two sample sets because it is a laborious task to validate all APIs and all (anti)patterns and because Facebook, Dropbox, Twitter, and YouTube are representative APIs [18]. Therefore, in the first phase, we choose one medium sized API, Dropbox, to calculate the recall on one API (the entire validation would have required 1,545 questions for 309 test methods).

In the second phase, we randomly selected 50 validation questions (out of 630 possible candidates) to measure overall accuracy. We used precision and recall to measure the detection accuracy. Precision is the ratio between the true detected

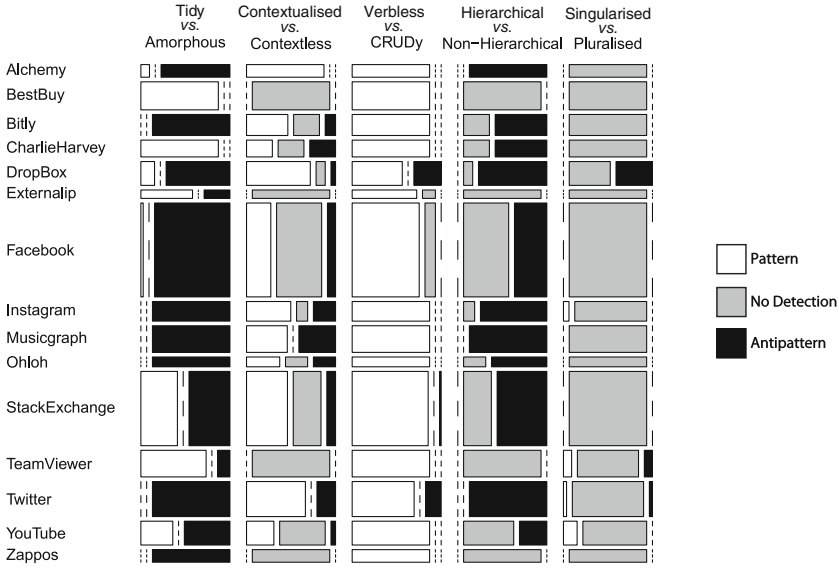


Fig. 4. Linguistic (anti)patterns detected in each RESTful API.

(anti)patterns and all detected (anti)patterns. Recall is the ratio between the true detected (anti)patterns and all existing true (anti)patterns.

5.4 Interpretation of the Results

The mosaic plot in Fig. 4 shows the pattern-wise representation of the detection results on the 15 RESTful APIs. Columns correspond to each (anti)pattern while rows represent the detected (anti)patterns on each API. In each row, the height of the mosaic represents the size of the method suite we tested for an API. In Fig. 4, the most frequent patterns are *Verbless URI* and *Contextualised Resource Names*—the majority of the analysed APIs did not include any CRUDy terms or any of their synonyms and the nodes in these URIs belong to the same semantic context. In contrast, the most frequent antipatterns are *Amorphous URI* and *Non-Hierarchical Nodes*—the majority of the analysed APIs involve at least one syntactical problem and that URI nodes for those APIs were not organised in a hierarchical manner. However, the conclusions drawn above are based on the analysis results obtained applying the DOLAR approach.

Table 3 presents detailed detection results for the ten linguistic (anti)patterns on 15 RESTful APIs. The table reports the (anti)patterns in the first column followed by the analysed RESTful APIs in the following fifteen columns. For each RESTful API and for each (anti)pattern, we report the total number of occurrences reported as positives by our detection algorithms. The last two columns show the total detected occurrences across 15 APIs (with percentage) and the average detection time. The detailed detection results for all the 309 tested

Table 3. Detection results of the ten REST lexical (anti)patterns (numbers in parenthesis show the number of methods tested for each API).

RESTful APIs	(9) Alchemy	(20) Bestbuy	(15) Bitly	(12) CharlieHarvey	(17) DropBox	(6) Externalip	(67) Facebook	(14) Instagram	(19) Musicgraph	(7) Ohloh	(53) StackExchange	(19) TeamViewer	(25) Twitter	(17) YouTube	(9) Zappos	(309) Total	Detection Time
Lexical Antipatterns/Patterns																	
✗Amorphous URI	8	0	15	0	14	2	65	14	19	7	28	3	25	10	9	219(71%)	0.984s
✓Tidy URI	1	20	0	12	3	4	2	0	0	0	25	16	0	7	0	90(29%)	0.968s
No Detection	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0(0.0%)	—
✗Contextless Resource Names	0	0	2	4	1	0	7	4	9	2	6	0	6	1	0	42(14%)	0.565s
✓Contextualised Resource Names	9	0	8	4	14	0	21	8	10	3	28	0	19	6	0	130(42%)	0.666s
No Detection	0	20	5	4	2	6	39	2	0	2	19	19	0	10	9	137(44%)	—
✗CRUDy URI	0	0	0	0	6	0	0	0	0	0	1	0	5	0	0	12(4%)	0.737s
✓Verbless URI	9	20	15	12	11	5	58	14	19	7	52	19	20	17	9	287(93%)	0.677s
No Detection	0	0	0	0	0	1	9	0	0	0	0	0	0	0	0	10(3%)	—
✗Non-hierarchical Nodes	9	0	10	8	15	0	28	12	19	5	34	0	25	6	0	171(55%)	0.584s
✓Hierarchical Nodes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0(0.0%)	0.592s
No Detection	0	20	5	4	2	6	39	2	0	2	19	19	0	11	9	138(45%)	—
✗Pluralised Nodes	0	0	0	0	8	0	0	0	0	0	0	2	1	0	0	11(4%)	0.668s
✓Singularised Nodes	0	0	0	0	0	0	0	1	0	0	0	2	1	3	0	7(2%)	0.656s
No Detection	9	20	15	12	9	6	67	13	19	7	53	15	23	14	9	291(94%)	—

methods from 15 RESTful APIs are available on our project Web site <http://sofa.uqam.ca/dolar/>.

As shown in Table 3, more than 70% of analysed URIs (219 out of 309) show amorphousness. Exceptionally, the Bestbuy API has all the URIs detected as *Tidy URI*. In contrast, all the URIs in Instagram and Twitter, for example, have syntactic problems and all of them are detected as *Amorphous URI*. As for the *Contextualised Resource Names* pattern, most of the APIs applied this pattern correctly—APIs providers use contextual resources names as nodes in URIs design—an important factor that affects the understandability of RESTful APIs. However, our dictionary-based analyses did not relate contexts among URI nodes for 137 cases because the dictionaries we used are general English dictionaries and do not relate to specific domains like social networks such as Twitter and Facebook. However, a domain specific dictionary might reason about URIs contexts more accurately.

We observe the same detection results for *Hierarchical Nodes* pattern, *i.e.*, the dictionaries could not find hierarchical relations among URIs nodes. Indeed, we have zero detection for *Hierarchical Nodes* pattern because: (1) around 50% of tested URIs used only one node (excluding the base URI) in which case we cannot check the hierarchical relation and (2) more than 20% URIs contain digits or numbers as nodes, which again do not fall under any hierarchical relations.

The occurrences of *CRUDy URI* antipattern were detected in only 4% (12 out of 309) tested URIs. In contrast, 93% (287 out of 309) of the tested URIs

are *Verbless URI*. In other words, APIs designers seem aware of not mixing the definition of traditional Web service operations and resource-oriented HTTP requests in REST. In traditional Web services, operation identifiers reflect what they are doing, whereas in REST, actions to be performed on a resource should be explicitly mentioned only using HTTP methods and not within a URI through a CRUDy term. Finally, there is a significant amount of *No Detection* for *Singularised vs. Pluralised Nodes* since about 90% of our tested requests used HTTP GET method. HTTP GET requests can retrieve both single and multitude of resources. However, for the remaining 10%, the *Pluralised Nodes* antipattern appeared more frequently than the *Singularised Nodes* pattern.

Here, we discuss the *Contextless Resource Names* antipattern in detail (since, it is our running example for this paper). Out of 309 tested URIs, 14% (42 occurrences) of them are detected as *Contextless Resource Names* antipatterns, 42% (130 occurrences) are detected as *Contextualised Resource Names* patterns, and 44% (137 occurrences) are detected as *None*. More specifically, for example, in Bestbuy, most of the URIs have only one node followed by parameters. We ignore parameters while we capture the context. Thus, if there is only one node in URIs, it is not possible to find any contextual relationship. Therefore, all the Bestbuy URIs are detected as *No Detection*.

In contrast, the Dropbox, Facebook, StackExchange, Twitter, and YouTube involve a high number of contextualised URIs naming. These good practices may help their APIs clients better understand and reuse. The following snippet shows two request URIs from Facebook where the URI nodes are considered to be in the same semantic context:

```
1. https://graph.facebook.com/v2.2/{user_id1}/mutualfriends/{user_id2}?access_token=CAATt8..
2. https://graph.facebook.com/v2.2/{user_id1}/friendlists?access_token=CAATt8..
```

For Facebook, our DOLAR approach reported 21 tested methods (out of 67) as *Contextualised Resource Names* patterns.

5.5 Further Discussion of the Results

Table 4 shows the validation results on Dropbox (Validation 1) and on four representative APIs (Validation 2). For the first validation, the average precision is 81.4% and recall is 78% for all (anti)patterns. For the second validation, the average precision is 79.7%.

In the first validation of Dropbox, two occurrences of *Verbless URI* are false positives. The terms ‘copy’ and ‘search’ (or their synonyms) were not considered CRUDy by our algorithm in `/1/copy_ref/dropbox/MyDropboxFolder/` and `/1/search/dropbox/MyDropboxFolder/`. However, the manual validation considered those terms CRUDy. Thus, on Dropbox, we had a precision of 100% and a recall of 75% for *CRUDy URI* and a precision of 80%, recall of 100% for *Verbless URI*.

The *Non-hierarchical Nodes* antipattern was detected by our detection algorithm in 14 cases whereas the manual validation suggested only three of them

Table 4. Complete validation results on Dropbox (Validation 1) and partial validation results on Facebook, Dropbox, Twitter, and YouTube (Validation 2). ‘P’ represents the numbers of detected positives and ‘TP’ the numbers of true positives.

Antipatterns/ Patterns	Validation 1						Validation 2					
	DOLAR		Validated	Precision	Average Precision	Recall	Average Recall	DOLAR		Validated	Precision	Average Precision
	P	TP						P	TP			
*Amorphous URI	13	12	12	92.31%	96.2%	100%	87.5%	4	4	4	100%	100%
✓Tidy URI	3	3	4	100%		75%		3	3	3	100%	
No detection	0	0	0	-		-		0	0	0	-	
*Contextless Resource Names	0	0	0	-	100%	-	100%	2	0	2	0%	53.3%
✓Contextualised Resource Names	14	14	14	100%		100%		5	3	5	60%	
No detection	2	2	2	100%		100%		3	3	3	100%	
*CRUDy URI	6	6	8	100%	90%	75%	87.5%	2	2	2	100%	100%
✓Verbless URI	10	8	8	80%		100%		9	9	9	100%	
No detection	0	0	0	-		-		0	0	0	-	
*Non-hierarchical Nodes	14	3	3	21.43%	60.7%	100%	66.7%	6	1	3	16.67%	58.3%
✓Hierarchical Nodes	0	0	11	-		0%		0	0	11	-	
No detection	2	2	2	100%		100%		4	4	5	100%	
*Pluralised Nodes	6	3	4	50%	60%	75%	48.3%	1	1	4	100%	86.7%
✓Singularised Nodes	0	0	2	-		0%		1	1	6	100%	
No detection	10	7	10	70%		70%		10	6	10	60%	
Average				Precision	81.4%	Recall	78%			Precision	79.7%	

actually are organised in a non-hierarchical order. We manually investigated the causes of such discrepancies, and found that the URIs that we identified as antipatterns by our detection algorithms and, later, were (manually) validated as patterns have the following URI pattern:

```

1. {baseURI}/{media|revisions|shares}/dropbox/MyDropboxFolder/...
2. {baseURI}/fileops/{copy|delete|move|create_folder}?root=dropbox&path=...
    
```

Our dictionary-based analyses did not find any hierarchical relations between {media, revisions, shares} and dropbox, between MyDropboxFolder and dropbox, and so on. Yet, these hierarchical relations are obvious for developers and it was easy to infer the hierarchical relations among those pairs simply because they use a natural naming scheme [11]. It is the same for the second example, where fileops and {copy, delete, move, create_folder} are validated to be in hierarchical relation and the English dictionaries could not find any hierarchical relations, thus DOLAR considered them as *Non-hierarchical Nodes* antipatterns. Therefore, for this antipattern, we had a low precision of 21.43%.

In the second validation, also for the *Non-hierarchical Nodes* antipattern, DOLAR faces a similar problem for Twitter as illustrated in these examples:

```

1. {baseURI}/help/privacy.json
2. {baseURI}/statuses/{show.json|user_timeline.json}?screen_name=...
    
```

The dictionary-based analyses did not find any hierarchical relations between ‘help’ and ‘privacy’ or between ‘statuses’ and {show, user, timeline} and reported them as non-hierarchical. The precision for *Non-hierarchical Nodes* antipattern is therefore 16.67%, due to this limitation with the analyses.

Finally, an interesting observation from Table 4: two cases were identified as *Contextless Resource Names* antipatterns that were manually validated as

Contextualised Resource Names pattern. Our investigation shows that the English dictionaries suggested ‘Canucks’ and ‘albums’ in Facebook and ‘followers’ and ‘list’ in Twitter to be in two different contexts. However, three professionals validated them as patterns, which caused the precision down to 0% for this antipattern in four representative APIs, with an average precision of 53.3%.

1. https://graph.facebook.com/Canucks/albums?access_token=CAA2...
2. https://api.twitter.com/1.1/followers/list.json?screen_name=...

5.6 Discussion on the Hypotheses

We now discuss the hypotheses defined in Sect. 5.1.

H₁. Accuracy: From Table 4, for the first validation on Dropbox API, we obtained an average precision of 81.4% and recall of 78% (Validation 1). As for the second validation, on a partial set of tested methods on Facebook, Dropbox, Twitter, and YouTube (*i.e.*, 50 out of 125 tested methods), we obtained an average precision of 79.7% (Validation 2). However, for the second validation, we cannot calculate recall because we validated only a part of all tested methods. Moreover, for the manually validated subset of URIs, we had a lower precision ranging between 16.67% and 21.43% only for *Non-hierarchical Nodes* antipattern due to the limitations of WordNet dictionary. Thus, despite lower precision for one specific antipattern, with an average precision of 81.4% and 79.7%, and a recall of 78% for all (anti)patterns, we can positively support our first hypothesis on the accuracy of our defined set of rules and the detection algorithms.

H₂. Extensibility: We added to SOFA ten new REST linguistic (anti)patterns, which required semantic analyses for their detection. At present, SOFA can detect a set of 23 REST (anti)patterns from both syntactic and semantic aspects. Furthermore, we added three new RESTful APIs (*i.e.*, Instagram, StackExchange, and Externalip), and more than 190 new HTTP requests from [18]. To add new (anti)patterns, one needs to implement and integrate their detection algorithms within SOFA architecture. To add a new RESTful API, one must add its service interface, the underlying methods of the service, an authenticated client that can invoke these methods, and a wrapper SCA component, which specifies the bindings, base URI, and various runtime properties. Thus, it is possible to add new (anti)patterns, which supports our second hypothesis.

H₃. Performance: Table 3 (last column) shows the detection time for each pattern and antipattern, ranging between 0.565 s and 0.984 s, with an average of 0.709 s. In fact, the total required time also includes the execution time, *i.e.*, sending requests and receiving responses (ranges from 2.074 s to 20.656 s, with an average of 6.92 s). We performed our experiment on an Intel Core-i7 with a processor speed of 2.50 GHz and 8 GB of memory. The reported detection times are comparatively low (on an average, 10% of the total required time). However, the total required time also depends on the number of tested methods for each

API. With such a low average detection time of 0.709s and execution time of 6.92s, we can positively support our third hypothesis on performance.

5.7 Threats to Validity

To minimise the threat to the *external validity* of our results, we performed experiments on 15 well-known APIs by invoking over 300 methods. We used WordNet for lexical and semantic analyses of URIs. However, one limitation of WordNet is that it does not include information on the semantic similarity between words. In addition, the number of defined relationships among words is limited and it lacks compound concepts or words. For example, we found URIs with compound resource identifiers that, when split, may cause losing contextual information. This threat to the *internal validity* affected our detection results. However, we plan to incorporate other similarity measure techniques like second order similarity to improve our semantic analysis.

The detection results may deviate depending on the defined algorithmic rules of linguistic (anti)patterns. Engineers may have their own views and levels of expertise on REST linguistic (anti)patterns, which may affect the definition of algorithmic rules. We tried to minimise this threat to the *construct validity* by defining all rules after a thorough review of definitions in existing literature on REST linguistic (anti)patterns. We also involved three professionals in the validation of the results and involved a third expertise if conflicts arose. Finally, to minimise the threat to *reliability validity*—the possibility to replicate this study—we gather the details to replicate this study, including the algorithmic rules and the client request URIs, on our Web site.

6 Conclusion and Future Work

REST client developers need to understand well RESTful APIs while designing and developing their own Web-based systems. Understandability and reusability are thus two major factors that APIs providers must consider. This paper presented DOLAR (Detection Of Linguistic Antipatterns in REST), an approach supported by the SOFA framework [17] extended with syntactic and semantic analyses, for the detection of linguistic (anti)patterns in RESTful APIs.

We applied DOLAR to specify ten linguistic (anti)patterns. We validated DOLAR by analysing 15 RESTful APIs and invoking 309 methods and showed its accuracy: (1) an average precision of 81.4% and recall of 78% on Dropbox and (2) an average precision of 79.7% for a partial validation on Facebook, Dropbox, Twitter, and YouTube. We also observed that out of the 15 analysed RESTful APIs, most of them involve syntactical URIs design problems and do not organise URIs nodes in a hierarchical manner. However, the REST APIs designers, in general, use appropriate resource names fit for a context and they do not use verbs in URIs, which is a good URIs design practice in REST.

As future work, we want to apply DOLAR on other RESTful APIs and to Open Linked Data. We plan also to include domain-specific ontologies in the

semantic analyses to overcome the limitations of English dictionaries and to apply other natural language processing techniques like second order similarities. We want to perform a validation of DOLAR results with RESTful APIs developers.

Acknowledgements. The authors thank Charlie Faucheu for initiating the study. This study is supported by NSERC (Natural Sciences and Engineering Research Council of Canada) and FRQNT, Canada research grants.

References

1. Abebe, S.L., Haiduc, S., Tonella, P., Marcus, A.: Lexicon bad smells in software. In: 2009 16th Working Conference on Reverse Engineering, pp. 95–99. IEEE (2009)
2. Arnaudova, V., Di, M.: Linguistic antipatterns: what they are and how developers perceive them. *Empirical Softw. Eng.* (2015)
3. Arnaudova, V., Eshkevari, L.M., Penta, M.D., Oliveto, R., Antoniol, G., Gueheneuc, Y.G.: REPENT: analyzing the nature of identifier renamings. *IEEE Trans. Softw. Eng.* **40**(5), 502–532 (2014)
4. Berners-Lee, T., Fielding, R.T., Masinter, L.: Uniform Resource Identifier (URI), Generic Syntax (2005)
5. Edwards, M.: Service Component Architecture (SCA). OASIS, USA, April 2011
6. Erl, T.: Service-Oriented Architecture: Concepts, Technology and Design. Pearson Education, Boston (2005)
7. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. Ph.D. thesis, University of California, Irvine (2000)
8. Fredrich, T.: RESTful Service Best Practices: Recommendations for Creating Web Services, May 2012. <http://www.restapitutorial.com/resources.html>
9. Goddard, C.: Semantic Analysis: A Practical Introduction. Oxford Textbooks in Linguistics, OUP Oxford (2011)
10. Hausenblas, M.: On entities in the web of data. In: Wilde, E., Pautasso, C. (eds.) REST from Research to Practice, pp. 425–440. Springer, New York (2011)
11. Laitinen, K.: Estimating understandability of software documents. *SIGSOFT Softw. Eng. Notes* **21**(4), 81–92 (1996)
12. Lawrie, D., Morrell, C., Feild, H., Binkley, D.: Effective identifier names for comprehension and memory. *Innovations Syst. Softw. Eng.* **3**(4), 303–318 (2007)
13. Lu, M., Sun, X., Wang, S., Lo, D., Duan, Y.: Query expansion via wordnet for effective code search. In: 22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering, Montreal, Canada, pp. 545–549 (2015)
14. Massé, M.: REST API Design Rulebook. O’Reilly, Sebastopol (2012)
15. Mateos, C., Rodriguez, J.M., Zunino, A.: A tool to improve code-first web services discoverability through text mining techniques. *Softw. - Pract. Experience* (2014)
16. Microsoft MSDN: Capitalization Styles. [https://msdn.microsoft.com/en-us/library/x2dbyw72\(v=vs.71\).aspx](https://msdn.microsoft.com/en-us/library/x2dbyw72(v=vs.71).aspx)
17. Moha, N., Palma, F., Nayrolles, M., Conseil, B.J., Guéhéneuc, Y.-G., Baudry, B., Jézéquel, J.-M.: Specification and detection of SOA antipatterns. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) ICSOC 2012. LNCS, vol. 7636, pp. 1–16. Springer, Heidelberg (2012)

18. Palma, F., Dubois, J., Moha, N., Guéhéneuc, Y.-G.: Detection of REST patterns and antipatterns: a heuristics-based approach. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) ICSSOC 2014. LNCS, vol. 8831, pp. 230–244. Springer, Heidelberg (2014)
19. Parrish, A.: Social Network APIs: A Revised Lexical Analysis (2010)
20. Rahman, M.M., Chanchal, R.K.: TextRank based search term identification for software change tasks. In: 22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering, Montreal, Canada, pp. 540–544 (2015)
21. Rodriguez, J.M., Crasso, M., Zunino, A., Campo, M.: Improving web service descriptions for effective service discovery. *Sci. Comput. Program.* **75**(11), 1001–1021 (2010)
22. Seinturier, L., Merle, P., Rouvoy, R., Romero, D., Schiavoni, V., Stefani, J.B.: A component-based middleware platform for reconfigurable service-oriented architectures. *Softw. Pract. Experience* **42**(5), 559–583 (2012)
23. Tilkov, S.: REST Anti-Patterns, July 2008. www.infoq.com/articles/rest-anti-patterns

Aggregating Functionality, Use History, and Popularity of APIs to Recommend Mashup Creation

Aditi Jain, Xumin Liu^(✉), and Qi Yu

B. Thomas Golisano College of Computing and Information Sciences,
Rochester Institute of Technology, Rochester, USA
{axj4268,xumin.liu,qi.yu}@rit.edu

Abstract. Creating mashups from existing Web APIs has provided an effective means to boost software reuse and approach the full potential of online programming resources. One of the key hindrance faced by mashup creation is to discover relevant APIs, especially due to the recent fast growth of Web APIs and the brief, unstructured API descriptions. In this paper, we propose a novel approach that recommends APIs to create a mashup given a free-form text description. We incorporate three heterogeneous but complimentary factors into the recommendation process: the functionality of an API, the usage history of the API by existing mashups, and the popularity of the API. We leverage probabilistic topic models to learn an API's functionality from its textual description and compute relevance between the API and the given mashup description. As most APIs lack a rich textual description, we extend the API discovery process by exploiting collaborative filtering to estimate the probability of an API being used by existing similar mashups. These two sources of information are then integrated through Bayes' theorem, which allows us to discover a set of functionally relevant APIs. The popularity of these APIs is then factored in to perform quality based ranking so that the best APIs can be recommended first. A comprehensive experimental study has been conducted on a real-world dataset to evaluate the efficiency and effectiveness of the proposed method. The result indicates that our method is efficient and provides better recommendation than other competitive methods.

1 Introduction

The advent and advance of the Web 2.0 paradigm have expanded the development of mashups and their use in various web and mobile applications. Web service mashup refers to the composition of several web services or APIs (as most of them are REST-ful) to augment the functionality of those APIs. For example, an application to show weather forecast on a map of a location may integrate the mapping API by Google and weather API by Weather Channel. Mashups let a developer reuse already existing APIs and save development time and provide higher quality and reliability with least effort [2]. As the interest in

developing mashups increases so does the number of APIs. For example, there are about 13,444 APIs as of May 2015 published on ProgrammableWeb. Selecting the best suited API for mashup creation is a strenuous task even for an experienced developer. Many of the APIs lie under the same umbrella of functionality, so picking an optimal API among them in terms of quality and user-interest further complicates the selection process. Other limitations in selecting services for mashup include the compatibility of services with each other, so analyzing the input/output parameters of the services is again laborious. Therefore, there's a need for an approach that could simplify the API selection process for mashup composition so that developers can focus on other parts of their applications.

Existing efforts on recommending services for mashup creation fall into three main categories that are based on functionality based, QoS, and social network, respectively. Each type of approaches focus on one aspect, making them vulnerable to the possible low quality and insufficient input of that particular type of information. Functionality based approaches focus on finding APIs that provide relevant functionality [6, 7, 9]. They leverage various information resources, such as structured and unstructured API descriptions, semantic markups, tags, topic models, and API categories, to identify those relevant APIs for a mashup. QoS-based approaches use QoS value as the main guidance for finding suitable APIs for a mashup [3]. The generation of a mashup is led by an optimization process, aiming to achieve the best QoS in the end. These approach require the input from users to identify those functionally related APIs. This could be very challenging for users given the huge number of available APIs online. Social network based approaches leverage the network among mashups, APIs, user etc. and then predict links for services and mashup [2, 10]. These approaches require user information as the input for API recommendation, which is difficult to obtain in many cases.

In this paper, we propose a novel approach that addresses the above limitations when recommending APIs to create a mashup given its textual description. The approach is hybrid as it estimates the recommendation probability of an API from three perspectives: functionality, usage history, and popularity. Specifically, our contributions in this work are summarized as follows.

1. We exploit **probabilistic topic models** to derive functional features of APIs and the desired mashup specification given by the user. Both the APIs and mashup can be then represented as probabilistic distributions on latent topics. The relevance of an API to a mashup is measured by the similarity between the topic distributions of its description and the mashup specification.
2. We leverage **matrix factorization based collaborative filtering** to identify additional functionally relevant APIs. The idea of using collaborative filtering is to leverage the mashup creation efforts made before, which were recorded in descriptions of existing mashups. As our goal is to create a mashup, no APIs have been used by such a new API yet, giving rise to the long-standing cold start issue in recommender systems. We instead focus on the existing mashups that are similar to the user desired mashup based on their topic distributions computed using probabilistic topic models. These

mashups are then used in a collaborative filtering algorithm to locate additional relevant APIs.

3. We **apply Bayes' theorem to integrate** the two sources of information obtained through probabilistic topic models and collaborative filtering, which gives the posterior probability of given an API being used by the new mashup. The top-k most probable APIs are identified, which are all considered as functionally relevant to the new mashup. The popularity of these APIs are then used to implicitly **perform QoS based ranking** and recommend the best ones to the user.

The remainder of this paper is organized as follows. In Sect. 2, we give an overview of existing effort that is relevant to the proposed approach. In Sect. 3, we present in detail the proposed API discovery and recommendation approach for mashup creation. In Sect. 4, we describe our experimental result. We conclude in Sect. 5.

2 Related Work

In this section, we discuss several representative related work and differentiate them with our work.

Functionality Based Recommendation. An approach was proposed to use Relational Topic Model (RTM) to identify functionally equivalent APIs for recommendation [6]. RTM determines topic distribution in a document considering all the citations and web links present in the document. The approach assumes that mashups and APIs form a network of documents where documents consist of topic related words and tags and links between them means that the API is part of the mashup. Now, to recommend a mashup, RTM model and binary random variables are used to predict the links. An iMashup tool was proposed to compose mashup based on a data-driven approach using tag-based semantic annotations [7]. It aims toward quick and easy composition of mashup as many end-users want ease in the discovery and integration of services to get the required final mashup. The tool makes use of tags to derive semantic annotations and links them to service's inputs/outputs. Then services are linked based on similar tags and a directed acyclic graph is constructed using those links. Depth first search and then regression search is run to obtain the recommendation services from the graph. In our work, we consider more factors besides functionality for recommendation. A category-based approach was proposed to identify the latent categories of potential APIs from the development requirement of a mashup and recommend the top ranked APIs in each category [9]. This method assumes that no more than one API should be selected from one category for a mashup, which is not always the case. As an example, a social mashup that allows users to access multiple social accounts may require several social APIs, such as Twitter, Facebook, and LinkedIn.

Quality-Based Recommendation. An approach was proposed to use the quality of APIs to drive the composition of a mashup [3]. A tool was developed to allow a user to drag APIs to add to a mashup. Then the tool calculates how the added API would influence the overall quality of the mashup and provide suggestions for alternative services. To start with the quality evaluation, first the QoS is taken into account and then the role of this service in the mashup is analyzed. Naturally, a master/central service would influence quality of the mashup more as compared to any slave/side services. This method requires a user to select the appropriate APIs from the functionality perspective, which imposes a great burden on users due to the large number of available APIs.

Social-Based Recommendation. A social-aware recommendation approach was proposed to address the implicit and explicit requirements of a user [10]. The approach explores and analyzes the social relationships between tags, mashups/API topics, and users by building a coupled matrix model. Then coupled-factorization algorithm is ran on the matrices obtained after the analysis to identify latent relationships and construct a recommendation prediction matrix. Another social-based approach was proposed by A. Maaradji et al. for service discovery and selection for mashup composition [8]. It proposed a platform called Social Composer (SoCo), which leverages social interactions of composition interests of users to derive mashup recommendations. It transforms the interactions between users and services to interactions between users. It uses an implicit social graph to map implicit social relations between users and links the users depending on their composition interest and activities. These links help build recommendation confidence based on the common interest of two users. The recommendation is based on analyzing user profiles for user's interests in services and relevance of services. An approach was proposed to solve the cold-start problem and under utilization of social information [4]. It makes use of both social and functional information to accelerate service discovery and recommendation for mashups. The method first extracts semantic descriptors from user's mashup query to discover candidate APIs and social features are extracted using popularity and collaboration ratings. Then, these semantic and social features of APIs are represented using graphs. The candidate mashup chains are assessed for input/output connectivity to recommend services for mashups. This approach assumes the knowledge of service input/output, which may be difficult to obtain for RESTful services that have become the majority of online APIs. In contrast, our approach can work with both structured and unstructured service descriptions so that it can be used for both SOAP based and RESTful services. The social based approaches request user information and the corresponding social network. Such information may be hard, or impossible to locate in many real-world scenarios.

3 The Proposed Approach

In this section, we describe in detail the proposed approach that aggregates API functionality, their usage history by existing mashups, along with their

popularity to recommend relevant APIs for mashup creation. The approach is composed of three interrelated components - (1) functionality based API discovery, (2) matrix factorization based collaborative filtering to expand the candidate API space, and (3) popularity based ranking for API recommendation.

Given the limited terms in the API descriptions, discovery of functionally relevant candidate APIs should go beyond simply matching terms in the descriptions. The recent advances in topic models in natural language processing and machine learning enable us to match APIs and mashups based on their underlying topics, which are expected to cover the functionality they provide. In particular, we exploit the Latent Dirichlet Allocation (or LDA [1]) model to determine the topics in API descriptions and the desired mashup specification given by the user. Then cosine similarity is used to determine the similarity between the topic distributions of API descriptions and the given mashup specification. Topic models still inherently rely on the use of terms to derive the underlying topics, based upon which the relevant APIs will be identified. As a result, the few terms used by most API descriptions may limit the effectiveness of topic models and hence relevant APIs may be missed out. To address this limitation, matrix-factorization based collaborative filtering is employed to discover additional candidate APIs based on the way they have been used in other existing mashups. We will focus on the existing mashups that are similar to the user desired mashup based on their topic distributions computed using LDA. Each of these two components will assign a relevance score to each API. Since the relevance scores lie in the range of 0 to 1, they can be interpreted as the probability of being relevant to the user desired mashup. By making the class conditional independence assumption, we can leverage the Bayes' theorem to integrate the two relevance scores into a single one, which allows us to choose a set of top-k candidate APIs. These top-k candidates will be finally ranked and returned to the user based on how popular they have been used in the past. As the top-k candidates are mainly determined from a functional perspective, the popularity of these APIs help factor in the non-functional aspects of these services, which enable to recommend functionally relevant services with good qualities.

In sum, the first two components of the proposed approach help discover a set of functionally relevant APIs for mashup creation while the last component uses popularity to implicitly perform QoS based ranking of these selected APIs and recommend the best ones to the user. Figure 1 illustrates the overall structure of the proposed approach.

3.1 Functionality-Based Candidate API Discovery

In this component, functionally relevant APIs are identified based on the given mashup specification in the mashup query. LDA is used to represent the functionality of available APIs and mashup specification by determining their topics.

LDA is a generative model, which regards each document as generated from a collection of topics and each topic is a distribution of words. It will determine the topic distribution in API descriptions and the mashup specification in the given mashup query to analyze their functionality. The API descriptions in this study are part of the API dataset from ProgrammableWeb.com that contains API

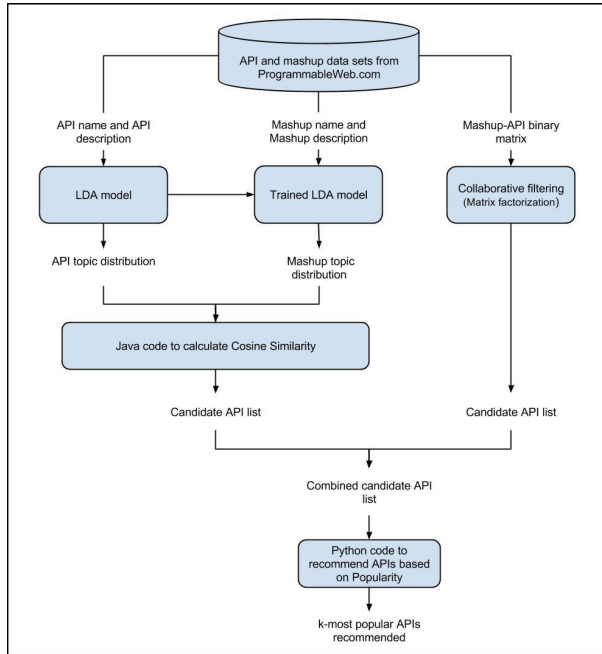


Fig. 1. Overall structure of the approach

names, descriptions of API, the number of mashups an API has been used for, API providers, and tags. Like many other model, LDA consists of two phases - learning or training phase and inference or testing phase. LDA is trained using API descriptions. In the training phase, the model takes all the API descriptions and the total number of topics as input. It tries to distribute words under different topics based upon their co-occurrences to obtain word-topic distributions. Meanwhile, it analyzes and assigns each document (or API description in this case) with probability of existence of each topic and gives topic probability distribution as the output. The topics having high probabilities contribute towards the identification of functionality of the API. In the testing phase, the topics for given mashup specification are inferred. The word-topic distributions are available from the training model obtained using API descriptions. The same topics from the training set are used to find the probability of those topics in the given mashup specification. After obtaining topic distributions for both APIs and mashup specification, the similarity between each API and the mashup specification can be calculated using the cosine similarity of their respective topic distribution vectors.

3.2 Historical Usage Based API Discovery

As stated above, the lack of rich API descriptions may limit the power of using topic models, such as LDA, to identify their underlying functionality and hence

discover APIs that are relevant to a user desired mashup. To address this issue, we propose to leverage historical usage of APIs in existing mashups and apply collaborative filtering techniques to identify additional candidate APIs.

Collaborative filtering is one of the most famous techniques for recommendation systems. It is based on the concept that if two users who previously preferred same items would prefer similar items in future. While recommending items to a user, all the users similar to an active user are found and the items preferred by those users are recommended. Alternatively, if a user prefers an item, similar items could be referred to the user. Thus, leveraging usage history of an item and a user, items could be recommended to users by analyzing the preference of users in past. Collaborative filtering could use two techniques - neighborhood-based or model-based. Neighborhood-based approaches usually use Pearson Correlation Coefficient to calculate the nearest or most similar users (and/or items) for recommendation. Though this technique is easy to understand and implement, it is usually not effective when the data is sparse and its performance decreases as the size of the dataset grows. On the other hand, model-based approaches, such as matrix factorization, are usually effective to overcome the data sparsity issue. These techniques are based on the idea that there are latent features/factors that could be discovered from user preferences and then be used for making recommendations [11].

Matrix factorization based collaborative filtering utilizes a user-item matrix, which usually represents user ratings for each item and may contain empty entries suggesting that user has not used that item and doesn't provide a rating for it. Thus, predicting the rating for these unused items would aid the recommendation process. This matrix is decomposed to learn latent factors and obtain two different matrices (corresponding to users and items, respectively) whose product recovers the original matrix. In practice, the original matrix is never recovered perfectly. The goal is to discover component matrices whose product minimizes the errors or differences between the original matrix and recovered matrix. After the errors are minimized, the new matrix would contain predicted approximate rating values for those empty entries. Early implementation of matrix factorization relies on Singular Value Decomposition (SVD), which is inefficient for large sparse data matrices. We instead exploit the Alternating Least Squares (ALS) method which works well with sparse data matrices and minimizes the squared errors by alternating between holding one of the factors fixed while computing the other [5].

The above idea can be applied to identify relevant APIs based on their usage history of existing mashups. Specifically, mashups play the role of users and APIs play the role of items. The key remaining issue is that since the goal is to create a new mashup, there is no usage history for the mashup yet. To still apply collaborative filtering, we instead seek for existing mashups that are similar to the desired mashup. We can again leverage LDA based topic models to determine the similarity between the new mashup specification and all existing mashup descriptions. In this way, the most similar existing mashups can be identified. After that, all the APIs used by these mashups will be included in the candidate list. This helps include some additional APIs but the number may still be limited

as most mashups only use a very small number of APIs. For example, after analyzing the mashups crawled from ProgrammableWeb.com, it was observed that around 85% of the mashups use only three or less APIs. We then use each of these mashups as if it is the new API and apply matrix factorization based collaborative filtering to identify more relevant APIs. Intuitively, APIs that have been used by mashups that are similar to new mashup stand a greater chance of being used again as they would be more contextually relevant and thus could be recommended for the required mashup. As the output, a list of candidate APIs for the new mashup is returned with a probability of the API being used by the new mashup.

3.3 Popularity Based API Ranking

By using topic models in the first component, each API is assigned a cosine similarity with the new mashup that a user desires to create. Similarly, by leveraging the API usage history through collaborative filtering in the second component, each API is assigned a probability of being used by the new mashup. In fact, the outputs from the two components can be both regarded as relevance scores of an API. Since the relevance scores take values in $[0, 1]$, we can interpret them as the probability of being relevant to the user desired mashup. Let $p(a_t|m)$ denotes the probability that API a is relevant to mashup m based on their topic distributions; let $p(a_u|m)$ denotes the probability that API a is relevant to mashup m based on its usage history in existing mashups similar to m . By assuming conditional independence, we can compute

$$p(a|m) = p(a_t|m)p(a_u|m) \quad (1)$$

$p(a|m)$ essentially specifies given a mashup m , how likely API a will be used in it. We are instead interested to know given an API a , how likely it will be used by the (new) mashup m , which is given by the posterior probability $p(m|a)$. By applying Bayes' theorem, we have

$$p(m|a) \propto p(a_t, a_u|m)p(m) = p(a_t|m)p(a_u|m)p(m) \quad (2)$$

$p(m)$ is the prior probability of observing a mashup m , which can be set to $1/M$ with M denoting the total number of mashups. We can use $p(m|a)$ to select the top-k most relevant APIs.

The top-k APIs are all considered as providing relevant functionality for the new mashup. However, some of these APIs may offer identical functionality. It would be desirable if these APIs can be ranked based on their nonfunctional properties (i.e., QoS) and returned to the user. Intuitively, APIs which have higher quality are the ones frequently used for constructing existing mashups, hence, they are more popular as compared to the others. Thus, the popularity score, which is computed as the number times an API has been used in existing APIs, can serve as a good QoS indicator and be used to rank the functionally relevant APIs. In this way, the recommended APIs are not only functionally relevant for the new mashup, but also provide a good QoS guarantee [6].

4 Experiments and Evaluation

We conducted a set of experiments to evaluate the efficiency and effectiveness of the proposed approach. We collected the experimental dataset by crawling the ProgrammableWeb.com, one of the largest public web API repository. We crawled the API and mashup profiles including their names, API categories, tags, brief descriptions, and the use of APIs by mashups. There are 10,325 APIs and 6,819 mashups in the dataset, but only the mashups that use four or more APIs were selected to get more tangible results. So, about 950 mashups were selected and used.

All experiments were carried out on a Macbook Pro with 2.6 GHz Core processor and 8 GB DDR3 memory under Mac OS X 10.9.5 operating system. The evaluation focus mainly on the accuracy of the recommendation. We use the summary of a mashup as the input and evaluate how well it can predict the APIs for the mashup, comparing the actual APIs used by the mashup with the ones recommended by our approach. We compare the proposed method with functionality based, collaborative filtering based recommendations, as well as some existing competitive methods [6].

4.1 Training Probabilistic Topic Models

We used the Mallet LDA package¹ to learn latent topics and the their probabilistic distributions for terms. As mashup descriptions are treated as the input from users that reflect their requirements on creating a mashup, such information is not always available before making recommendations. Therefore, we only uses API descriptions for the topic learning process. The performance of LDA is affected by the predefined number of topics. The optimal number can be obtained through a set of trials and the observation on the resulted word-topic distribution for each trial. If many semantically unrelated words are assigned to the same topic, then the specified number of topics should be bigger. On the other hand, if many semantically related words are assigned to different topics, the specified number of topics should be smaller. Following this guidance, we trained the model with different number of topics, starting from 30 to 100 with an interval of 5 topics. We settled at 65 topic as this gave us the most readable result. The output of the LDA model training are topic-document and word-topic distributions. Figure 2 illustrates distribution of 14 topics over 10 different API descriptions. Figure 4 visualizes some example words for topics 4, 63, and 45. The font size of each word in the table is proportional to the probability of the word appearing in that topic. For example, the word *interface* appears more number of times as compared to word *developer* in topic 45 which corresponds to *web-related* topic. The relation between APIs *flickr* and *tumblr* can be seen through Figs. 2 and 4 that topic 63 has the highest probability, which consists of words like *social*, *friends*, *photos* etc. (which we know is true). Also, Fig. 3 illustrates percentage of API descriptions assigned to different topics. It shows

¹ <http://mallet.cs.umass.edu/>.

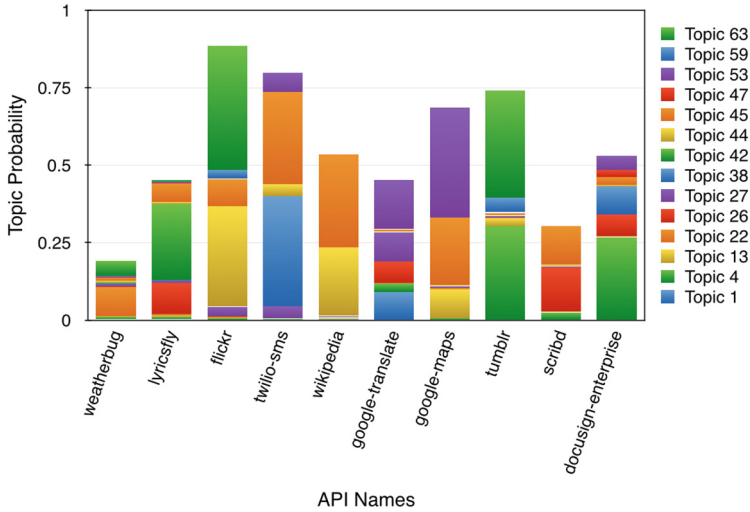


Fig. 2. Stacked bar graph illustrating topic distribution over API description

that topic 45 is assigned to approximately 21% of the API descriptions. This is because it contains words like *interface* and *protocol*, which are common words used for API descriptions.

After training the LDA model, the model was used to infer topics in mashup description of the mashup query. To infer the topics, the word-topic distribution is piped from the model trained with API description. We use the same topics from the training set and find the probabilities of those topics in mashup description. The inference output of the mashup query is the list of topics with their probabilities. The topics in each document are ordered based on their topic probabilities. Hence, the first topic assigned for each document is the most prominent topic and so on.

4.2 Evaluation Result

We used the current API used by mashups as the ground truth to evaluate the accuracy of our recommendation method. We divided the mashups into three sets: training, validation, and testing. In the training and validation sets, the use of APIs by mashups is assumed to be known. The testing set consists of 302 mashups, where the use of APIs by these mashups were compared to the recommendation result for the evaluation. We compared our recommendation method with two other ones: topic modeling based and collaborative filtering based approaches. As most of the existing mashup recommendation methods do not tackle the cold start problem, their experimental results are not comparable to ours except for the one proposed in [6]. Therefore, we compared our method to it.

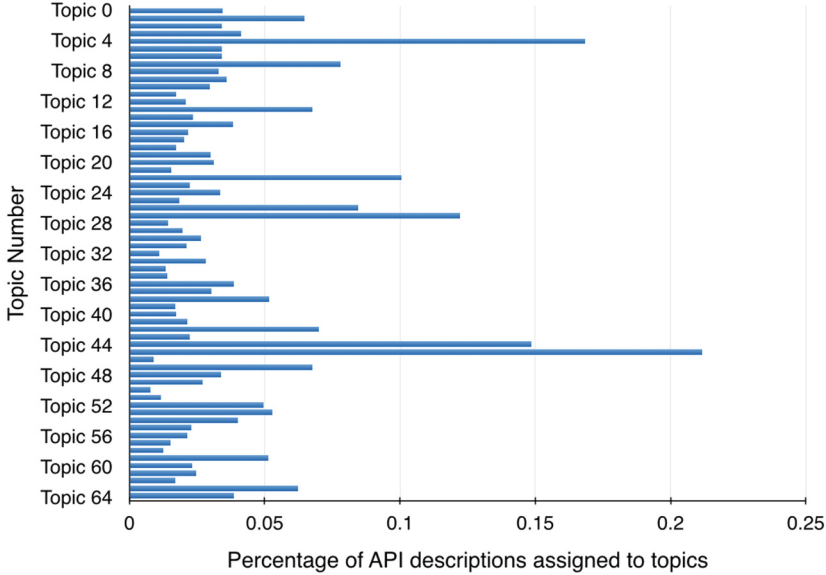


Fig. 3. Bar graph illustrating percentage of API descriptions assigned to topics

Traditional metrics for measuring prediction accuracy of a recommendation system includes *precision* and *recall*. Precision is the ratio of the total number of properly recommended APIs to the total number of recommended APIs. Recall is the ratio of the total number of properly recommended APIs to the total relevant APIs. In this work, we use $recall@T$ to examine the impact of the number of recommended APIs on the accuracy. That is, given a constant T , a recommendation system suggests the top T APIs. We set T from 10 to 100, with an interval of 10. Recall is calculated as the ratio of the total number of properly recommended APIs in the suggested list (N_{T_i}) to the total number of APIs actually used in the mashup (N_{m_i}). Therefore, the overall performance of the recommendation is the average of all recalls evaluated in the testing set, i.e.,

$$recall@T = (1/M) \sum_{i=1}^M (N_{T_i}/N_{m_i}) \quad (3)$$

We also use F-score as an evaluation metric. F-score integrates both recall and precision by computing the harmonic mean of them as follows, i.e.,

$$F\text{-score} = \frac{Recall * Precision}{Recall + Precision}$$

Figure 5 shows the average recall values for top- T recommended APIs and compares each of the approach where $T \in \{10, 20, \dots, 100\}$. It can be seen that the proposed hybrid recommendation method outperforms the LDA based and

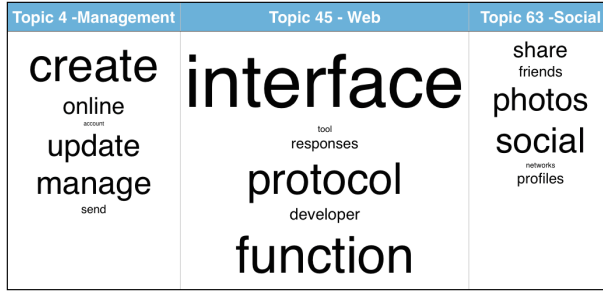


Fig. 4. Example of word-topic distribution

collaborative filtering method methods. On the one hand, LDA can find similar APIs, but doesn't consider mashup's usage history, which leads to recommending APIs that are not relevant or not preferred by the mashup. This explains why it performs poorly. On the other hand, collaborative filtering considers mashup usage history and can recommend APIs that are preferred by the mashups, thus performing much better than LDA. However, collaborative filtering suffers from the cold-start problem. Therefore, combining both approaches to give a hybrid solution seems logical to overcome these issues to some extent. Moreover, integrating popularity score of APIs adds QoS factors in the recommendation as usually most popular APIs are acknowledged as being of high quality. Figure 5 also demonstrates that the value of recall increases as the number of recommended APIs increases for all the approaches. This is a logical behavior as the probability of relevant APIs being recommended would be higher when more APIs are recommended. But when this number is higher than 80 the recall is almost constant and when recommended APIs lie between 60–80 there is not much difference in recall values. This indicates that about 60–80 APIs could be recommended in order to achieve similar performance and recommending less number of APIs would reduce mashup creator's further workload.

Table 1. Comparison of proposed approach with the ERTM Approach in [6]

	Recall	Average precision
Proposed approach	0.62	0.41
ERTM approach	0.27	0.16

Table 1 compares our work and the one proposed in [6]. In order to make a fair comparison, we used the metric in their work to evaluate the accuracy. That is, average precision gives the accuracy of ranked recommendation list i.e. it considers the order of the recommended list and computes if the results in higher rank/position in the list are relevant or not. This measure is computed as it is usually desirable in search results to know if top results are relevant to the user or not. It was computed as:

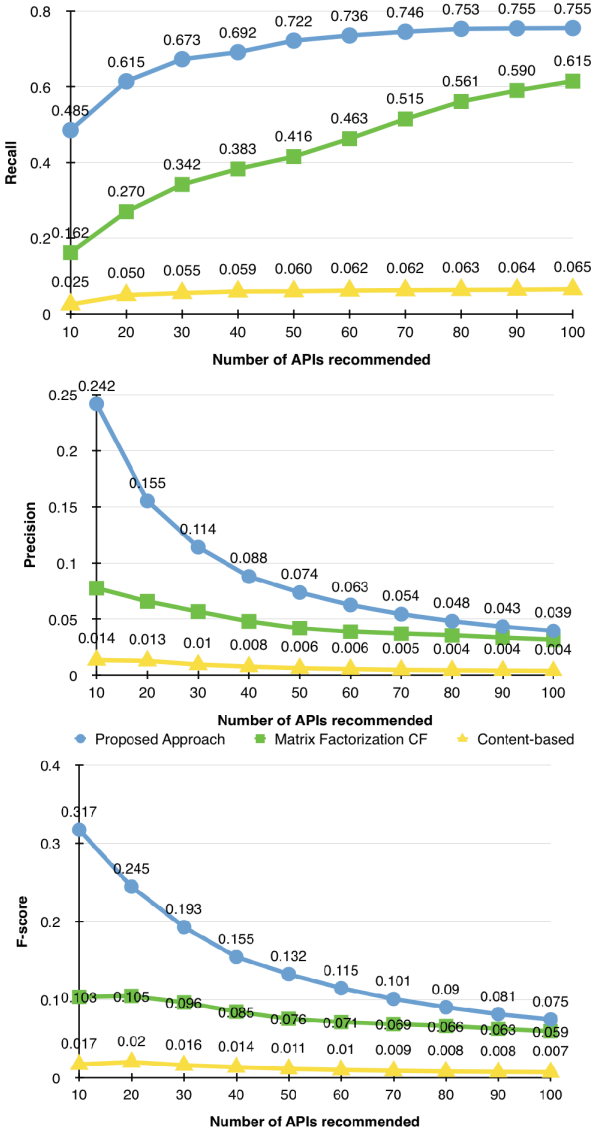


Fig. 5. Average Recall, Precision and F-score comparison for the proposed approach, individual matrix factorization CF and LDA with cosine similarity. The number of recommended APIs was increased to analyze its effect on the performance.

$$Average\ Precision = \frac{\sum_{k=1}^n (P(k) \times rel(k))}{|Relevant\ APIs|}$$

where k is the rank/position of the API in the list; $P(k)$ is the precision at cut-off k and $rel(k)$ is the change in recall or in simple words, it indicates if the API at

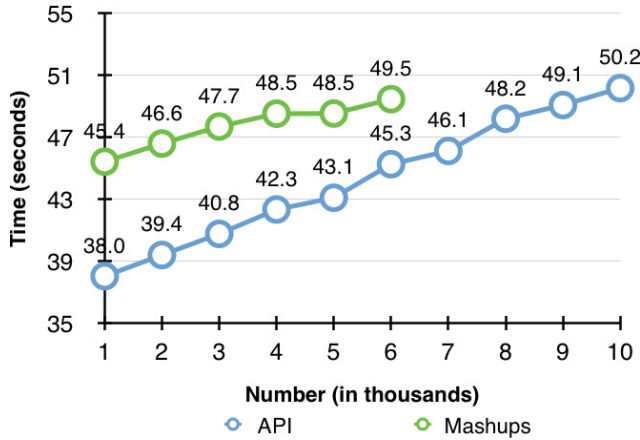


Fig. 6. Influence of increasing number of APIs and mashups on matrix factorization’s performance

rank k is relevant or not. Its value is 1 if the API is relevant, otherwise it is 0. As shown in Table 1, the average recall of our approach is 0.62, which is better than the one in [6]. Our approach achieves a better precision as well, i.e., 0.27 vs 0.16.

Finally, to evaluate the performance of the approach, performance of matrix factorization phase was evaluated. This is because its performance is directly proportional to the increasing number of APIs and mashups as it increases data density. Also, performance of LDA depends on the number of topics and first phase could be performed offline, therefore, it is not considered in the evaluation of performance of the complete approach. Figure 6 shows that time required to train the matrix factorization model and predict ratings increases with the increase in the number of APIs and mashups.

5 Conclusion

In this paper, we propose a novel approach to recommend relevant APIs with good QoS for mashup creation. The proposed approach integrates functionality and usage history to discover functionally relevant APIs and then uses their popularity in existing mashups to achieve a ranked list of candidate APIs. By aggregating multiple sources of information, the proposed approach helps discover a set of functionally relevant APIs for mashup creation while performing QoS based ranking of these selected APIs so as to recommend the best ones to the user. The results of the experiments demonstrate that the proposed approach outperforms both individual matrix factorization and content-based (LDA plus cosine similarity) approaches. Moreover, it also has better accuracy as compared to other competitive methods.

Our future work will focus on improving recommendation accuracy by exploring advanced topic modeling techniques and checking the orchestration compatibility of candidate APIs.

Acknowledgments. This work was supported by US National Science Foundation under grant DUE-1141200.

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. *J. Mach. Learn. Res.* **3**, 993–1022 (2003)
2. Cao, B., Liu, J., Tang, M., Zheng, Z., Wang, G.: Mashup service recommendation based on user interest and social network. In: 2013 IEEE 20th International Conference on Web Services (ICWS), pp. 99–106, June 2013
3. Cappiello, C., Matera, M., Picozzi, M., Daniel, F., Fernandez, A.: Quality-aware mashup composition: issues, techniques and tools. In: 2012 Eighth International Conference on the Quality of Information and Communications Technology (QUATIC), pp. 10–19, September 2012
4. Jung, J., Lee, K.-H.: Socially-enriched semantic mashup of web APIs. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) *Service Oriented Computing. LNCS*, vol. 7636, pp. 389–403. Springer, Heidelberg (2012)
5. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. *Computer* **8**, 30–37 (2009)
6. Li, C., Zhang, R., Huai, J., Sun, H.: A novel approach for API recommendation in mashup development. In: 2014 IEEE International Conference on Web Services (ICWS), pp. 289–296, June 2014
7. Liu, X., Zhao, Q., Huang, G., Mei, H., Teng, T.: Composing data-driven service mashups with tag-based semantic annotations. In: 2011 IEEE International Conference on Web Services (ICWS), pp. 243–250, July 2011
8. Maaradji, A., Hacid, H., Skraba, R., Lateef, A., Daigremont, J., Crespi, N.: Social-based web services discovery and composition for step-by-step mashup completion. In: 2011 IEEE International Conference on Web Services (ICWS), pp. 700–701, July 2011
9. Xia, B., Fan, Y., Tan, W., Huang, K., Zhang, J., Wu, C.: Category-aware API clustering and distributed recommendation for automatic mashup creation. *IEEE Trans. Serv. Comput.* **PP**(99), 1 (2014)
10. Xu, W., Cao, J., Hu, L., Wang, J., Li, M.: A social-aware service recommendation approach for mashup creation. In: 2013 IEEE 20th International Conference on Web Services (ICWS), pp. 107–114, June 2013
11. Yu, Q.: Cloudrec: a framework for personalized service recommendation in the cloud. *Knowl. Inf. Syst.* **43**(2), 417–443 (2015)

Integrating Gaussian Process with Reinforcement Learning for Adaptive Service Composition

Hongbing Wang¹(✉), Qin Wu¹, Xin Chen¹, and Qi Yu²

¹ School of Computer Science and Engineering and Key Laboratory
of Computer Network and Information Integration,
Southeast University, Nanjing, China

hbw@seu.edu.cn, {bellawu627,cyceve}@gmail.com

² College of Computing and Information Sciences,
Rochester Institute of Tech, Rochester, USA
qi.yu@rit.edu

Abstract. Service composition offers a powerful software paradigm to build complex and value-added applications by exploiting a service oriented architecture. However, the frequent changes in the internal and external environment demand adaptiveness of a composition solution. Meanwhile, the increasingly complex user requirements and the rapid growth of the composition space give rise to the scalability issue. To address these key challenges, we propose a new service composition scheme, integrating gaussian process with reinforcement learning for adaptive service composition. It uses kernel function approximation to predict the distribution of the objective function value with strong communication skills and generalization ability based on an off-policy Q-learning algorithm. The experimental results demonstrate that our method clearly outperforms the standard Q-learning solution for service composition.

1 Introduction

In service computing, when a single web service can not meet a complex user requirement, combining multiple existing services to build a complex value-added service becomes a common practice, leading to services composition [6]. However, network-based web services are inherently dynamic. Therefore, a particular composition solution may become infeasible before execution due to the changes in the internal and external service composition environment (e.g., Quality of Service or QoS declining or functional decay). Therefore, a composition solution needs to adapt to those uncertain factors and deliver an adaptive and reliable composition solution to users [19]. In addition, the complexity of a composition workflow and the growth of candidate services lead to a large composition space, which can be expressed by m^n with m being the number of abstract service in a composition workflow and n being the number of candidate services for each abstract service [4, 16]. Given the above challenges, we should provide a new

composition solution, which achieves certain adaptability while addressing the scalability issue at the same time.

Adaptive service composition as a hot topic attracts attentions and recent studies mainly use integer programming, graph planning, reinforcement learning and so on. Among them, integer programming may be limited by the scale of the problem. Graph planning is poorly suited to a dynamic environment. Existing reinforcement learning methods are also falling short for large-scale problems [20]. For a large-scale and dynamic service composition problem, multi-agent, hierarchical reinforcement learning and function approximation technologies provide some promising directions, some of which have been applied into services composition with some success. For example, our previous work addressed the problem by exploiting multi-agent technologies [20] and achieved a relatively good composition performance. In this paper, we aim to explore function approximation techniques to deal with large-scale service composition as the proposed composition solution is built upon an reinforcement learning algorithm. Reinforcement learning concerns the problem of a learning agent interacting with its environment to achieve a given goal [1]. Instead of being given examples of desired behavior, the learning agent must discover by trial and error how to behave in order to get the most reward, which means that reinforcement learning methods have inherent adaptability for a dynamic environment [20]. However, table-based reinforcement learning algorithms, such as the Q-learning algorithm [1], only perform well in small-scale problems. They lack the generalization ability for large-scale problems.

Function approximation techniques overcome the drawbacks of exact representations for value functions and policies in reinforcement learning algorithms. They can solve problems in large or continuous state and action spaces [2]. In addition, function approximations [2] can be separated into two main types: parametric and nonparametric. Parametric approximations map from a parameter space into the space of functions with predetermined forms and number of parameters. The parameters are tuned using training data about the target function. Unlike the parametric case, nonparametric approximation also has parameters, but the number of parameters is determined from the data instead of a prior. Thus nonparametric approximations are more flexible for the practical and large problems, where it is hard to predefine the number of parameters.

In this paper, we propose a new adaptive composition solution using an off-policy reinforcement learning algorithm integrated with gaussian process, a nonparametric approximator. GP is a kind of Bayesian Nonparametric (BNP) function approximation model. In the large-scale service composition framework, we first model the service composition problem with a Markov decision process, then utilize an off-policy Q-learning algorithm to achieve the optimal or near-optimal composition scheme. In order to adapt to the large-scale scenarios, we model the Q-value function evaluation process with a kernel function nonparametric approximator to improve the composition performance. Our contributions are summarized as follows:

- We introduce the MDP-WSC (Markov Decision Process-Web Service Composition) model to address large-scale service composition in a dynamic and complex environment.

- We optimize a reinforcement learning algorithm based on gaussian process for service composition. It is a kernel function approximation technique and can predict the distribution of the objective function value with strong communication skills and generalization ability.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 introduces the problem formulation and basic definitions. Section 4 presents our approach for service composition based on off-policy Q-learning integrated with gaussian process. In Sect. 5, some experimental results are presented for evaluating the proposed approach. The paper is concluded in Sect. 6.

2 Related Work

In this section, we review some existing works that are most relevant to our approach, including reinforcement learning (RL) and gaussian process (GP) adopted in service composition.

Wang et al. [23] proposed an adaptive RL method based on a Markov Decision Process (MDP) that finds an optimal solution at runtime. A MDP builds a model for obtaining compositions consisting of multiple aggregated workflows. The work in [20] extended the ideas in [23] and proposed a optimized model for service composition in multi-agent scenarios. Liu et al. [14] proposed an improved RL approach that utilizes the reuse strategy to enhance performance and stability of RL techniques. However, they impose high computational cost especially in large service environments. Moustafa et al. [15] proposed an approach to the QoS-aware service composition problem using multi-objective reinforcement learning. But the method is not very efficient for large-scale service composition scenarios.

The above RL methods for service composition are based on a look-up table, which is difficult to extend to a large-scale scenario. An possible solution is to replace the look-up table and value function with function approximation techniques [3]. Most function approximation techniques can be classified as parametric and non-parametric approximation [2]. GP is one of the common non-parametric function approximation techniques [18]. It is a natural generalization of multivariate gaussian random variables to infinite (countably or continuous) index sets. GP has been applied in a large number of fields to a diverse range of ends, and many deep theoretical analyses of various properties are available. It is attractive because of its flexible non-parametric nature and computational simplicity [17].

Yaakov Engel [8] proposed an on-line learning approach to the problem of value function estimation in continuous state spaces by imposing a gaussian prior over value functions and assuming a gaussian noise model. They also proposed a SARSA based extension that allows gradual improvement of policies in [9]. Jonathan Ko [13] presented a general technique for system identification that combines GP and RL into a single formulation, which is done by training a GP on the residual between the non-linear model and the ground truth training data.

Thomas Gartner [11] investigated the use of GP to approximate the quality of state-action pairs and employed GP in relational RL by using graph kernels as the covariance function between state-action pairs.

3 Problem Formulation

Similar to our previous studies [20–23], we use a Markov Decision Process (MDP) to model the selection of services to form a service composition.

Definition 1 (MDP-Based Web Service Composition (MDP-WSC)).
 A MDP-WSC is a 6-tuple $MDP-WSC = \langle S, S_0, S_\tau, A(\cdot), P, R \rangle$, where

- S is a finite set of world states;
- $S_0 \in S$ is the initial state from which an execution of the service composition starts;
- $S_\tau \subset S$ is the set of terminal states, indicating an end of composition execution when reaching one state $S_\tau^i \in S_\tau$;
- $A(s)$ represents the set of services that can be executed in state $s \in S$;
- P is the probability distribution function. When a web service α is invoked, the world makes a transition from its current state s to a succeeding state s' . The probability for this transition is labeled as $P(s' | s, \alpha)$;
- R is the immediate reward function. When the current state is s , and a service α is selected, then we can get an immediate reward $r = R(s, a)$ from the environment after executing an action.

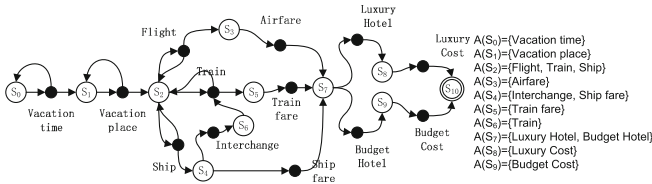


Fig. 1. The MDP-WSC of a Composite Service

Figure 1 shows a MDP-WSC graph of a composite service for a vacation plan. It consists of two kinds of nodes, i.e., state nodes and service nodes, which are represented by open circles and solid circles, respectively. s_0 is the initial state node, and nodes with double circles are terminal state nodes, such as s_{10} . A state node can be followed by a number of invoked service nodes, labeled with the transition probability $P(s' | s, \alpha)$. Immediate reward r can be expressed by aggregated QoS value of a service [23]. A MDP-WSC transition graph can be created by using some automatic composition approaches, such as an AI planner [16]. In addition, a MDP-WSC model has sufficient expression ability to describe a business process control flow [23], and its solution is a deterministic policy π , which determines the service selection under the specified state.

There are many reinforcement learning algorithms for solving MDP problems, from which the off-policy Q-learning algorithm is a widely used strategy for its simplicity and insensitivity for policy [24]. The off-policy Q-learning algorithm incrementally evaluates actions' Q-values according to the reward function and Q-function. Its iterative formula can be seen in Eq. (1), where α is the learning rate and γ is the discount factor.

In our large-scale service composition scenario, our solution may face poor performance if we directly apply the off-policy Q-learning algorithm to search the optimal (or near-optimal) composition sequence since the Q-learning algorithm is limited by the solution space. Therefore, we need to optimize the off-policy Q-learning algorithm to adapt to the large-scale composition scenario.

$$Q(s, a) \leftarrow (1 - \alpha) * Q(s, a) + \alpha * (r + \gamma * \max_{a'} Q(s', a')) \quad (1)$$

Function approximation is designed to address large-scale or continuous state space problems [2]. In particular, kernel-based nonparametric approximation, which is supported by a lot of statistical literature, directly tune parameters from observed data without specifying prior parameter forms and numbers. It can achieve more accurate characterization of the state space and is more suitable for online learning. Therefore, it is widely applied in machine learning algorithms [7, 17]. For a deeper understanding of the principles of kernel methods, we first introduce several important definitions and theorems related to kernel function approximation.

Definition 2 (Reproducing Kernel Hilbert Space). H denotes a real-value Hilbert function space defined in an abstract set X , $\forall f(x) \in H, x \in X$, if there is a binary function $k : X \times X \rightarrow \mathfrak{R}$, which satisfies the following conditions:

1. For any fixed $y \in X$, $k(x, y) \in H$ as a function of x .
2. For any $f \in H$, $f(y) = \langle f(\cdot), k(\cdot, y) \rangle_H$.

Then, kernel k is called a reproducing kernel of H , H is called reproducing kernel space for k , abbreviated as RKHS.

We give the Representer Theorem based on Reproducing Kernel Hilbert Space, which is the theoretical basis of kernel methods.

Theorem 1 (Representer Theorem). Suppose X is a non-empty set, $k(\cdot, \cdot)$ is a positive definite real-valued kernel for $X \times X$, and also is the reproducing kernel for Hilbert space H_k . Given a sample set $(x_1, y_1), \dots, (x_n, y_n) \in X \times \mathfrak{R}$, a strictly increasing real-valued function $g : [0, \infty] \rightarrow \mathfrak{R}$, and any risk function $R : (X \times \mathfrak{R}^2)^m \rightarrow \mathfrak{R} \cup \{\infty\}$, then the following objective function $f^* \in H_k$ satisfies:

$$f^* = \arg \min_{f \in H_k} \{R((x_1, y_1, f(x_1)), \dots, (x_n, y_n, f(x_n))) + g(\|f\|)\},$$

and f^* satisfies the following equation,

$$f^*(\cdot) = \sum_{i=1}^n \theta_i k(\cdot, x_i), \text{ for any } 1 \leq i \leq n, \theta_i \in \mathfrak{R}.$$

According to the Representer Theorem, the value function of a reinforcement learning algorithm can be expressed as Eq. (2), where s denotes observed data, s_i denotes samples, θ denotes parameter vector, and $k(\cdot, \cdot)$ denotes the kernel function. Another theorem having a profound impact on kernel methods is Mercer theorem, and it also provides theoretical support for the widely used kernel trick.

$$V(s) = \sum_{i=1}^n \theta_i k(s, s_i). \quad (2)$$

Theorem 2 (Mercer Theorem). *Let $k(\cdot, \cdot)$ be a positive-definite, symmetrical, continuous and bounded kernel function. Then, the (positive-definite) integral operator $T_k : \Theta \rightarrow \Theta$ defined by $T_k \theta(x) = \int_x k(x, x') \theta(x') \rho_x(x') dx'$, where $\rho_x(\cdot)$ is the marginal distribution of x , has a countable set of continuous eigenfunctions $\{\psi_i\}_{i=1}^{\infty}$ with their respective positive eigenvalues $\{\lambda_i\}_{i=1}^{\infty}$, such that for almost all x, x' , $k(x, x') = \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(x')$.*

According to Mercer theorem, if we define $\phi_i = \sqrt{\lambda_i} \psi_i$, then $k(x, x') = \sum_{i=1}^{\infty} \phi_i(x) \phi_i(x') = \phi(x)^T \phi(x')$, and we can get an useful corollary, referred to as “Kernel Trick”.

Corollary 1 (Kernel Trick): Any algorithm, which may be stated using only inner products between members of the input space, can be immediately replaced with a new (kernel) algorithm, in which the inner products are replaced with kernel evaluations.

An algorithm, applying Kernel Trick to convert to a non-linear kernel form, will be understood as a linear algorithm in the feature space. For example, according to the kernel expression of Representer Theorem $\sum_{i=1}^t \alpha_i k(x_i, x)$, after using the Kernel Trick, we can get the expression $w^T \phi(x)$, where $w = \sum_{i=1}^T \alpha_i \phi(x_i)$.

In this paper, we utilize gaussian process, a Bayesian nonparametric (BNP) function approximation kernel method, to model Q-value function evaluation in the algorithm learning process so as to address large-scale service composition problem. Gaussian process [3, 5, 17], seeking to a maximum posterior probability through Bayesian inference, can achieve a probability distribution of Q-value for a reinforcement learning algorithm, which is helpful for state-space search in the learning process. Specifically, it is defined as the following:

Definition 3 (Gaussian Process). *Gaussian process can be seen as a set of random variables, wherein each random variable contains an input variable x ($x \in X, (x \in \mathbb{R}^d)$), and for any finite random variables f_x are subject to a joint Gaussian distribution.*

$$f \sim gp(m, k) \quad (3)$$

A gaussian process can be uniquely determined by the mean function $m(x) = E(f_x)$ and covariance function $k(x, x') = E[(f_x - m(x))(f_{x'} - m(x'))]$, wherein k is the kernel function. Under the noisy environment, given the training sample input and the corresponding output value $\{(x_i, f_i) | i = 1, \dots, n\}$, f_* is output value corresponding to the testing input set X_* . Then we can get a joint distribution,

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim N \left(0, \begin{bmatrix} K(X, X) + \omega_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix} \right) \quad (4)$$

where the $K(X, X_*)$ denotes $n \times n_*$ (n is the number of training sample, n_* is the number of testing points) covariance matrix between training samples and testing points. $K(X, X)$, $K(X_*, X)$, and $K(X_*, X_*)$ are similarly defined. ε denotes noise, $y(x) = f(x) + \varepsilon$, and $\text{cov}(y) = K(X, X) + \omega_n^2 I$.

Then, we can get the posterior predictive equation of a noisy gaussian process as following:

$$\begin{aligned} f_* | X, y, X_* &\sim N(\bar{f}_*, \text{cov}(f_*)), \\ \bar{f}_* &\triangleq E[f_* | X, y, X_*] = K(X_*, X) [K(X, X) + \omega_n^2 I]^{-1} y, \\ \text{cov}(f_*) &= K(X_*, X_*) - K(X_*, X) [K(X, X) + \omega_n^2 I]^{-1} K(X, X_*). \end{aligned} \quad (5)$$

4 Reinforcement Learning for Service Composition Based on Gaussian Process

4.1 Predicting Q-Value Based on Gaussian Process

When modeling the Q-value function with a gaussian process, the corresponding input field is all state-action pairs, and the desired result is a function of the distribution of Q values. More specifically, $Z = [z_1, \dots, z_\tau]$ represents the sample collection of observed action-state pairs, and a action-state pair is labelled as $z = \langle s, a \rangle$. $\vec{y} = [y_1, \dots, y_\tau]^T$ is the observed value vector corresponding to the sample collection of action-state pairs. Given some data points \vec{y} for the input field Z , we aim to predict the value of Q-function $y_{\tau+1}$ at new input point $z_{\tau+1}$. We use $K(Z, Z)$ to represent the kernel matrix, and take the corresponding $K_{l,m} = k(z_l, z_m)$ as the covariance between state-action pair z_l and z_m . $K(Z, z_{\tau+1})$ indicates the estimation of the kernel vector for state $\tau + 1$. ω_n^2 represents the possibility of uncertainty for estimation. Based on Eq. (5), we can derive the estimation and covariance of Q-value

$$\begin{aligned} \hat{Q}(z_{\tau+1}) &= m(z_{\tau+1}) = \alpha_\tau^T K(Z, z_{\tau+1}), \\ \text{cov}(z_{\tau+1}) &= k(z_{\tau+1}, z_{\tau+1}) + \omega_n^2 - K^T(Z, z_{\tau+1}) [K(Z, Z) + \omega_n^2 I]^{-1} K(Z, z_{\tau+1}) \end{aligned} \quad (6)$$

where $\alpha_\tau = [K(Z, Z) + \omega_n^2 I]^{-1} \mathbf{y}$.

We use this as the updating formula of the Q-learning algorithm, and replace $Q(s, a)$ with the estimated value $\hat{Q}(s, a)$ according to the gaussian posterior prediction.

$$\hat{Q}(s, a) = (1 - \alpha) * \hat{Q}(s, a) + \alpha * (r + \gamma * \max_{a'} \hat{Q}(s', a')) \quad (7)$$

We update \hat{Q} according to the newly observed data. The accuracy of observation depends on the accuracy of the current model. ω_n^2 , which represents the gaussian noise, serves as a regularization item here. It can prevent the model from converging too fast to an inaccurate estimation Q^* .

4.2 Constructing the Sparse Dictionary Online

Although integrating gaussian process with reinforcement learning can improve the flexibility and accuracy, the continually increasing sample space in iterative processes may lead to an increase for computational complexity at a polynomial rate (the time complexity is usually $O(\tau^3)$, and τ is the size of trained sample data), which may cause thorny challenges to practical use. Given this, there is a need for sparsification of the sample space, aiming at constructing a sparse dictionary and thus reducing the number of redundant samples and speeding up the convergence.

The method for dictionary construction can be classified as either on-line or off-line. Off-line dictionary construction uses either feature selection or feature extraction methods. Kernel principal component analysis (KPCA) is a common method, which is an extension of the standard PCA by exploiting the kernel trick. The computational complexity of standard feature decomposition using KPCA is $O(n^3)$.

The basic idea of online dictionary construction is as following: if a new sample z_i can be converted to a linear representation by samples in the dictionary, then the new sample will not join in the dictionary. Assuming that at $t - 1$, we get a sample dictionary, $D_{t-1} = \{\phi(z_1), \phi(z_2), \dots, \phi(z_{M_{t-1}})\}$, where $\phi(z_i)$ is the feature vector of z_i in the dictionary D and M is the size of the dictionary (i.e., $M = |D_{t-1}|$). Online dictionary construction methods include Approximate Linear Dependence (ALD), Projection and Novel Criterion (NC). Due to the online requirement of reinforcement learning, we need to construct a sparse dictionary online to guarantee the effectiveness and efficiency. Approximate Linear Dependence (ALD), which finds approximate answers for full rank conditions, has been used in reinforcement learning. It can construct a sparse dictionary online according to the condition of approximate linear dependence.

For a new feature vector $\phi(z_t)$, the condition of approximate linear dependence can be depicted as following:

$$\delta_t = \min_c \left\| \sum_j c_j \phi(z_j) - \phi(z_t) \right\|^2 \leq \xi \quad (8)$$

where $c = [c_j]$ and ξ is the threshold that determines the approximation quality and sparsity. When the condition in Eq. (8) is satisfied, the feature vector $\phi(z_j)$ will be ignored. Otherwise, it will join in the sample set.

$$\begin{aligned}
 \delta_t &= \min_c \left\| \sum_j c_j \phi(z_j) - \phi(z_t) \right\|^2 \\
 &= \min_c \left\{ \sum_{i,j} c_i c_j \langle \phi(z_i), \phi(z_j) \rangle - 2 \sum_i c_i \langle \phi(z_i), \phi(z_t) \rangle + \langle \phi(z_t), \phi(z_t) \rangle \right\},
 \end{aligned} \tag{9}$$

By using the kernel trick, $\langle \phi(x), \phi(y) \rangle = k(x, y) = k_{xy}$, we can derive that

$$\delta_t = \min_c \{ c^T K_{t-1} c - 2c^T k_{t-1}(z_t) + k_{tt} \}, \tag{10}$$

where the solution of Eq. (10) is

$$\begin{aligned}
 c_t &= K_{t-1}^{-1} k(t-1)(z_t), \\
 \delta_t &= k_{tt} - k_{t-1}^T(z_t) c_t
 \end{aligned} \tag{11}$$

We can see that, the computation of every step in the ALD method is mainly focusing on getting a inverse of the kernel matrix. So, the computational complexity is $O(M^2)$, and M is the size of the sample dictionary. We use the method mentioned above to construct a sparse dictionary.

4.3 Updating the Gaussian Process Parameters

We know that the dictionary is the basis of prediction by a gaussian process. The functional form and parameters of a gaussian process are updated by data samples in the dictionary. Now we will introduce the method for updating the gaussian process parameter, which is similar to what in [5].

Given a dictionary Z_d , according to Eq. (6), the predictive value and covariance are computed as following:

$$\begin{aligned}
 m(z_{\tau+1}) &= \alpha_\tau^T k(Z_d, z_{\tau+1}) \\
 cov(z_{\tau+1}) &= k(z_{\tau+1}, z_{\tau+1}) + k^T(Z_d, z_{\tau+1}) C_\tau k(Z_d, z_{\tau+1})
 \end{aligned} \tag{12}$$

where $C_\tau = -(K + \omega_n^2 I)^{-1}$. Given a new data point, the kernel matrix transposition and weight α can be computed according to the rank of the current kernel matrix. When updating online, we first give the definition of the following scalars:

$$\begin{aligned}
 q^{\tau+1} &= \frac{y - \alpha_\tau^T k_{x_\tau}}{\omega_n^2 + k^T(Z_d, z_{\tau+1}) C_\tau k(Z_d, z_{\tau+1}) + k(z_\tau, z_\tau)}, \\
 r^{\tau+1} &= -\frac{1}{\omega_n^2 + k^T(Z_d, z_{\tau+1}) C_\tau k(Z_d, z_{\tau+1}) + k(z_\tau, z_\tau)}
 \end{aligned} \tag{13}$$

We take $e_{\tau+1}$ as unit vector, the operators $\mathbb{T}_{\tau+1}(\cdot)$, $U_{\tau+1}(\cdot)$ means expanding the τ dimensional vector and matrix to $\tau + 1$ dimensional vector and matrix (by adding in 0). Consequently, the gaussian process parameter can be computed recursively by the following equations:

$$\begin{aligned}
 \alpha_{\tau+1} &= \mathbb{T}_{\tau+1}(\alpha_\tau) + q^{\tau+1} s_{\tau+1}, \\
 C_{\tau+1} &= U_{\tau+1}(C_\tau) + r^{\tau+1} s_{\tau+1} s_{\tau+1}^T, \\
 s_{\tau+1} &= \mathbb{T}_{\tau+1}(C_\tau k_{x_{\tau+1}}) + e_{\tau+1}.
 \end{aligned} \tag{14}$$

4.4 OGPQ Algorithm

In this section, we introduce the main steps of Q-learning for large-scale service composition based on a gaussian process (referred to as OGPQ). By constructing the sparse dictionary online as well as predicting the distribution of Q-values and updating the gaussian process parameter, we can derive an algorithm for large-scale service composition based on kernel methods. The algorithm is given below.

```

1: Initialization: discount rate  $\gamma$ , learning rate  $\alpha$ ,  $\hat{Q}(s, a) = 0 (s \in S, a \in A)$ ,
   initial state  $s_0$ , terminal state  $s_r$ ,  $BV = \{\}$ 
2: repeat
3:   for every episode, every time step  $\tau$  do
4:     According to  $\epsilon$ -greedy policy, select the service  $a_\tau$ 
5:     execute the service  $a_\tau$ , observe the reward  $r_\tau, s_{\tau+1}$ , make  $z_\tau = \langle s_\tau, a_\tau \rangle$ 
6:      $y_\tau = r + \gamma \max_{a_{\tau+1}} \hat{Q}(s_{\tau+1}, a_{\tau+1})$  (equal to 7)
7:     if  $\delta_{\tau+1} > \xi$  (judge by the ALD method) then
8:        $BV = BV + z_\tau$ 
9:     end if
10:    compute  $K_{z_{\tau+1}}, \alpha_{\tau+1}$ 
11:    update  $\hat{Q}(z_{\tau+1}) = m(z_{\tau+1}) = \alpha^T K(Z, z_{\tau+1})$ 
12:    Until  $s_{\tau+1}$  is the terminal state
13:  end for
14: until the convergence condition is satisfied

```

Algorithm 1. OGPQ Algorithm

BV in the dictionary represents the sample dictionary, which is empty at first. After the selection of services according to the ϵ -greedy policy, we can judge whether a new sample should join the dictionary based on the newly observed data (state-action input z_τ and the corresponding output y_τ). Then we update the parameter of the gaussian process and adjust the predicted Q-value, until the algorithm converges to an optimal estimation Q^* .

5 Experiments and Analysis

In this section, we present the experimental result of our proposed service composition method. We demonstrate the effectiveness, adaptivity and scalability of the off-policy Q-learning algorithm integrated with a gaussian process. We also compare it with the standard Q-learning algorithm, and analyze the results.

5.1 Experiment Setting

We randomly generate MDP-WSC transition graphs and use them as the input, and choose four QoS attributes from the extended QWS Dataset¹, which are

¹ <http://www.uoguelph.ca/~qmahmoud/qws/>.

ResponseTime, Throughput, Availability and Reliability. A number of key parameters are set up for the experiments as follows. The learning rate α of the standard Q-learning algorithm (referred to as GRQ in what follows) is set to 0.6 according to the study in [23]. In this paper, the learning rate for the Q-learning algorithm integrated with a gaussian process (referred to as OGPQ in what follows) is $\frac{300}{300+n(s)} * \beta$ ($\beta = 0.6$, $n(s)$ is the number of accessing state s), which is a form of $\alpha = \frac{a}{b+k}$ satisfying the convergence condition of iterative algorithm [10, 12]. The discount factor γ is set to 0.9 and the ϵ -greedy exploration strategy value is set to 0.6. Other parameters are set as follows: recency regulators factor $\mu = 0.1$, gaussian kernel $k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma_k^2}\right)$, $\sigma_k = 0.1$, threshold parameter of sparsity $\xi = \frac{\pi}{2}$, regularization term $\omega_n^2 = 1$. The experiments are conducted on an Intel i3-2120 3.30 GHz PC with 4 GB RAM.

5.2 Result Analysis

1. Validation of Effectiveness. The purpose of the first experiment is to examine the effectiveness of OGPQ compared with GRQ. In this scenario, an MDP-WSC has 100 state nodes and 100 candidate services for each state. We can see from Fig. 2(a), OGPQ is obviously superior to GRQ with regard to convergence rate, where OGPQ converges at about the 3000th episode and GRQ converges at about the 3600th episode. Since OGPQ performs online training, its performance is not outstanding in initial learning, but with more and more training samples, the gaussian posterior value prediction tends to be mature, which helps guide the state-space search and accelerate convergence. In contrast, GRQ can not effectively utilize the learning experience to guide learning process. It performs a random exploration, which is limited by the large-scale composition space. Thus, its convergence rate is relatively slower. In addition, OGPQ also achieves a higher cumulative reward value than that of GRQ, which means that OGPQ is closer to the optimal composition solution than GRQ.

Overall, this experiment verifies the effectiveness of OGPQ. It also demonstrates its superiority in terms of exploration and convergence when compared with GRQ.

2. Validation of Adaptability. The purpose of the second experiment is to verify the adaptability of OGPQ. The setting of the service state nodes and candidate services is the same with the first experiment. To simulate a dynamic environment, we randomly change the QoS values from a fixed number of candidate services during the learning process. In order to facilitate comparison, we change the QoS after the 1500th learning episode and before the 1600th learning episode. According to the experimental results in Fig. 2(b), OGPQ and GRQ both finally achieve convergence in spite of the dynamic environment, which demonstrates the adaptability of both algorithms. In addition, OGPQ is superior to GRQ pertaining to convergence rate and discount cumulative reward value. Since OGPQ predicts the Q-value distribution based on samples, the QoS fluctuation in the learning process has little effect on convergence rate and the discount cumulative

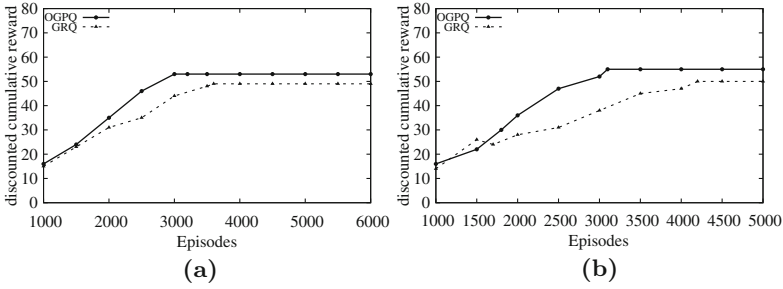


Fig. 2. (a) Validation of effectiveness (b) Validation of adaptability

reward value. It converges at about the 3100th learning episode with discount cumulative reward value of 55. In contrast, GRQ is totally dependent on the composition learning algorithm, which needs to relearn the optimal composition solution and delay convergence when the QoS of candidate services changes.

In sum, this experiment verifies the adaptability of the proposed OGPQ algorithm facing with a dynamic composition environment, which is beneficial to provide a more reliable service composition solution.

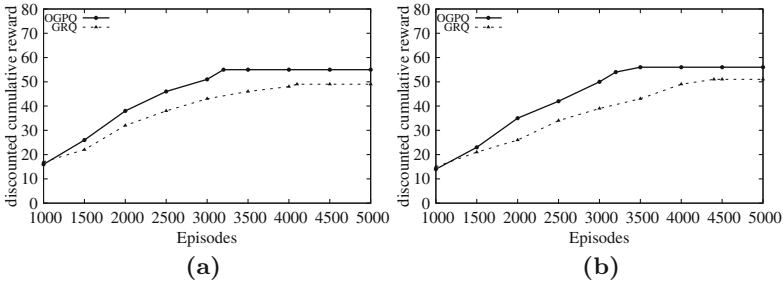


Fig. 3. (a) 200 candidate services, (b) 300 candidate services

3. Validation of Scalability. Here, we examine the influence of the states and candidate services respectively to verify the scalability of the proposed OGPQ algorithm. Firstly, we vary the number of services for each state node from 200 to 500 while fixing the state nodes at 100. From Figs. 3 and 4, we can see that OGPQ always has a distinct advantage, and converges at about the 3200th episode, 3400th episode, 3800th episode and 4000th episode, respectively when the candidate services for each state increasing from 200 to 500. On the other hand, GRQ converges at about the 4100th episode, 4400th episode, 4800th episode and 56000th episode, respectively. That is to say, GRQ can not converge before the 5000th episode in the 500 candidate services scenario. GRQ is a pure table-based learning algorithm and its learning efficiency will drop rapidly when

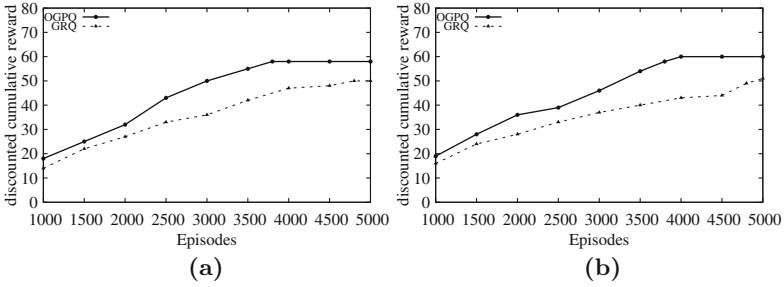


Fig. 4. (a) 400 candidate services, (b) 500 candidate services

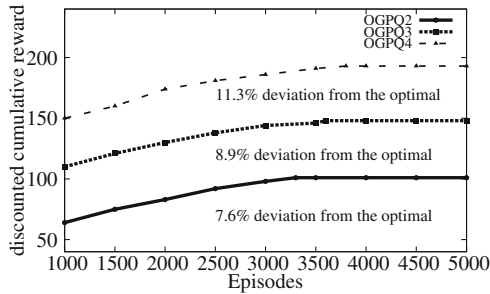


Fig. 5. Different state number for OGPO

facing a large-scale composition scenario. The proposed OGRP algorithm, which integrates the generalization ability of a gaussian process, can address large-scale problems. Thus it still maintains a strong scalability in a large-scale service composition scenario.

Next, we fix the candidate service number as 100 for each state node and increase the state nodes from 200 to 400, to explore the impact of the growth on state nodes for service composition. As the number of state nodes directly affects the discount cumulative reward value, we use deviation degree D to perform experimental analysis. Deviation degree is given by $D = \frac{OPR - CCR}{OPR}$, where OPR indicates the optimal convergence reward, and CCR is the current convergence reward. We can see from Fig. 5, the deviation degree D of OGPO in the scenarios of 200 state nodes, 300 state nodes and 400 state nodes is 9.8%, 15.1%, and 20.5%, respectively. The more state nodes the higher of D . That is to say, the increasing number of state nodes may result in more deviation from the optimality and fall into local optima. Hence, we can conclude that the OGPO has the scalability when face with the increment of states nodes.

To sum up, the OGPO algorithm can be applied to large-scale service composition scenarios with good scalability compared with GRQ.

6 Conclusions and Future Directions

To take QoS into consideration and maintain composition adaptivity and efficiency in a large-scale scenario, we propose a service composition approach based on reinforcement learning and gaussian process. In our approach, we first model the service composition problem using a MDP-WSC model, and aggregate all the QoS values into the reward function. In this way, the optimal service composition problem is transformed into a stochastic decision process problem. We then use the modified Q-learning algorithm to compute the solution, which integrates with a gaussian process. Through experimental analysis, we have demonstrated the effectiveness, adaptivity, and scalability in large-scale service composition.

The proposed approach can be further improved from the following aspects:

- In our framework, we assume that the environment can be observed fully, which may be not practical in some complex scenarios. To overcome this, a more generalized decision model based on Partially Observed Markov Decision Process can be introduced in the future work.
- The ALD method used to achieve the sparseness of the online dictionary still faces the problem of efficiency. We will try to exploit the NC method (whose time complexity is $O(n)$), which may reduce the computational complexity.
- The size of QWS dataset used in our experiment can not meet our requirements for large-scale service composition scenarios. We plan to collect more real services' information and thus to construct a large-scale service dataset for service composition.

Acknowledgments. This work is partially supported by NSFC Key Projects (No. 61232007 and 61532013) and Doctoral Fund of Ministry of Education of China (No. 20120092110028)

References

1. Barto, A.G.: Reinforcement Learning: An Introduction. MIT press, Cambridge (1998)
2. Busoniu, L.: Reinforcement learning and dynamic programming using function approximators. In: Automation and Control Engineering Series (2010)
3. Carl Edward Rasmussen, M.K.: Gaussian processes in reinforcement learning. Adv. Neural Inf. Process. Syst. **16**(2004), 751–759 (2004)
4. Constantinescu, I., Faltings, B., Binder, W.: Large scale, type-compatible service composition. In: Proceedings of the IEEE International Conference on Web Services (ICWS), pp. 506–513. IEEE (2004)
5. Csató, L., Opper, M.: Sparse on-line gaussian processes. Neural Comput. **14**(3), 641–668 (2002)
6. Dustdar, S., Schreiner, W.: A survey on web services composition. Int. J. Web Grid Serv. **1**(1), 1–30 (2005)
7. Engel, Y.: Algorithms and representations for reinforcement learning. Ph.D. thesis, Citeseer (2005)

8. Engel, Y., Mannor, S., Meir, R.: Bayes meets bellman: the gaussian process approach to temporal difference learning. In: ICML, vol. 20, p. 154 (2003)
9. Engel, Y., Mannor, S., Meir, R.: Reinforcement learning with gaussian processes. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 201–208. ACM (2005)
10. Even-Dar, E., Mansour, Y.: Learning rates for Q-learning. *J. Mach. Learn. Res.* **5**, 1–25 (2004)
11. Gärtner, T., Driessens, K., Ramon, J.: Graph kernels and gaussian processes for relational reinforcement learning. In: Horváth, T., Yamamoto, A. (eds.) ILP 2003. LNCS (LNAI), vol. 2835, pp. 146–163. Springer, Heidelberg (2003)
12. Gosavi, A.: A tutorial for reinforcement learning. Department of Engineering Management and Systems Engineering (2011)
13. Ko, J., Klein, D.J., Fox, D., Haehnel, D.: Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In: 2007 IEEE International Conference on Robotics and Automation, pp. 742–747. IEEE (2007)
14. Liu, Q., Sun, Y., Zhang, S.: A scalable web service composition based on a strategy reused reinforcement learning approach. In: 2011 Eighth Web Information Systems and Applications Conference (WISA), pp. 58–62. IEEE (2011)
15. Moustafa, A., Zhang, M.: Multi-objective service composition using reinforcement learning. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 298–312. Springer, Heidelberg (2013)
16. Oh, S.C., Lee, D., Kumara, S.R.: Effective web service composition in diverse and large-scale service networks. *IEEE Trans. Serv. Comput. (TSC)* **1**(1), 15–32 (2008)
17. Rasmussen, C.E.: Gaussian processes for machine learning (2006)
18. Taylor, G., Parr, R.: Kernelized value function approximation for reinforcement learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 1017–1024. ACM (2009)
19. Trummer, I., Faltings, B.: Optimizing the tradeoff between discovery, composition, and execution cost in service composition. In: Proceedings of the IEEE International Conference on Web Services (ICWS), pp. 476–483. IEEE (2011)
20. Wang, H., Chen, X., Wu, Q., Yu, Q., Zheng, Z., Bouguettaya, A.: Integrating on-policy reinforcement learning with multi-agent techniques for adaptive service composition. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) ICSOC 2014. LNCS, vol. 8831, pp. 154–168. Springer, Heidelberg (2014)
21. Wang, H., Wang, X.: A novel approach to large-scale services composition. In: Ishikawa, Y., Li, J., Wang, W., Zhang, R., Zhang, W. (eds.) APWeb 2013. LNCS, vol. 7808, pp. 220–227. Springer, Heidelberg (2013)
22. Wang, H., Wu, Q., Chen, X., Yu, Q., Zheng, Z., Bouguettaya, A.: Adaptive and dynamic service composition via multi-agent reinforcement learning. In: Proceedings of the IEEE International Conference on Web Services (ICWS), pp. 447–454. IEEE (2014)
23. Wang, H., Zhou, X., Zhou, X., Liu, W., Li, W., Bouguettaya, A.: Adaptive service composition based on reinforcement learning. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 92–107. Springer, Heidelberg (2010)
24. Wiering, M., Van Otterlo, M.: Reinforcement learning. In: Wiering, M., van Otterlo, M. (eds.) *Adaptation, Learning, and Optimization*, vol. 12. Springer, Heidelberg (2012)

Scalable SaaS-Based Process Customization with *Case Walls*

Yu-Jen John Sun^(✉), Moshe Chai Barukh, Boualem Benatallah,
and Seyed-Mehdi-Reza Beheshti

School of Computer Science and Engineering, University of New South Wales,
Sydney, Australia

{johns,mosheb,boualem,sbeheshti}@cse.unsw.edu.au

Abstract. The rising popularity of SaaS allows individuals and enterprises to leverage various services (e.g. Dropbox, Github, GDrive and Yammer) for everyday processes. However, these disparate services do not in general communicate with each other, rather used in an ad-hoc manner with little or no customizable process support. This inevitably leads to “shadow processes”, often only informally managed by e-mail or the like. In this paper, we propose a framework to simplify the integration of disparate services and effectively build customized processes. The implementation of the proposed techniques includes an agile services integration platform, called: *Case Walls*. We provide a knowledge-based event-bus for unified interactions between disparate services, while allowing process participants to interact and collaborate on relevant cases.

1 Introduction

Traditional structured process-based systems increasingly prove too rigid amidst today’s fast-paced and knowledge-intensive environments. A large portion of processes, commonly described as “unstructured” or “semi-structured” processes, cannot be pre-planned and likely depend upon human-interpretation. On the other hand, there has been a plethora of apps to support everyday tasks with enhanced collaboration. For example, *Software-as-a-Service (SaaS)* based tools such as: (i) *Dropbox* to store and share files; (ii) *Pivotal tracker* to manage tasks and projects; and (iii) *Google Drive* to edit and collaborate. Workers often need to access, analyze, as well as integrate data from various such cloud data services.

Albeit, there are crucial gaps in the SaaS-enabled endeavor: The large number of available services do not easily communicate with each other - often employed ad-hoc. Moreover, such ready-made services implies conforming to a fixed set of embedded features allowing little or no room for customization. Alternatively, even if a collection of such services are used for different portions of tasks, this inevitably leads to “shadow processes”, where synchronization between such services is handled in an ad-hoc manner (e.g., actions are often accomplished in a number of non-traceable steps via manual tasks, such as email or the like).

At the same time, the intent of the SOA-based approach was to simplify service integration via APIs. However, this was met by the inherent need to

understand various low-level APIs, leading to inflexible and costly programming environments, requiring multiple and continuous patches. To counteract this, advances in process composition languages emerged (e.g. BPEL), along with Mashup environments (e.g. Yahoo! Pipes). Albeit, these environments rarely provide productivity support tools akin to modern IDEs (e.g. code search and discovery, ease of reuse, debugging, code generation). Moreover, they suffer from the lack of agility and cannot support run-time changes. We thus call for an exciting new change: where advanced techniques for *simple, declarative, flexible exploration* and *manipulation* of multiple services in large scale and dynamic environments, are needed. We argue, the ubiquity of process and services will have little value if users cannot use, share and reuse them simply.

To address the above challenges, in this paper we propose a framework to *simplify* service-integration and effectively build customized processes. Central is the notion of *Case Knowledge Graph (CKG)*, where common services-related low-level logic can be abstracted, incrementally *shared* and thereby *reused* by developers. We organize knowledge into various dimensions: *APIs, Resources, Events* and *Tasks*. By identifying entities (i.e. attributes and relationships, along with their specialization), a novel foundation is introduced to accumulate dispersed case knowledge in a structured manner. This offers a unified representation, manipulation and reuse of case knowledge to empower simplified SaaS-enabled process customization. Empowered by this knowledge graph, we provide a novel case customization and deployment platform, called *CaseWalls*, which enables:

- Professional process developers to incrementally create modular collections of tasks - reusable and customizable process fragments (referred to as a “*case*”). E.g. Create an issue on a project management service; upload a file into a document management service; send an email when a co-worker upload new version of a file; post videos and photos into social media services, etc.
- A simple, declarative yet powerful language that allow case-workers to search existing tasks and compose into customized definitions. Composite tasks are abstracted as reusable cases in the CKG for further reuse. The tasks search component uses a “*context*” to describe the task “*intent*” and “*objective*” (e.g., upload a file, create an issue). Thus, using the business scenario to query the CKG can return tasks that are appropriate for the given context.
- An event/activity “*wall*” to inform case-works about task progress; together with a simple and declarative language to enable such participants to uniformly and collectively react, interact and collaborate on relevant case.
- The above is supported by a unified, knowledge-based event-bus for case orchestration. The knowledge required at runtime to orchestrate cases (i.e., detecting events, executing tasks, invoking APIs) is automatically extracted from the defined case definitions and expressed as event-action case orchestration rules. (We thus reuse a rule-engine as our execution environment.)

The rest of this paper is organized as follows: In Sect. 2, we propose a unified case knowledge-graph and describe the constituent entities and relationships. In Sect. 3, we present a novel knowledge-driven and declarative case manipulation

language. In Sect. 4, we present our implementation; evaluation in Sect. 5; then, related-work and conclusions in Sect. 6.

2 Case-Knowledge Representation and Reuse

A knowledge-graph (KG) is an effective technique for taxonomy-based organization of concepts and relationships. For example, Google-KG¹ is a graph of popular *informational* concepts on the Web, such as “people, places and things”. In our work, we apply a KG to curate (and thereby enable reuse) of service and process related programming concepts (i.e. *APIs*, *Operations*, *Resources*, *Events*, *Tasks*) and their relationships. Formally, a KG is defined as an ordered pair $G = (V, E, A_v, A_e)$ where, V is the vertex set whose elements are the vertices (i.e. nodes or entities) of the graph; E is the edge set whose elements are the edges (or relationships) between vertices of the graph; A_v is a set of attributes that can be associated with any vertex in V ; and A_e is a set of attributes that can be associated with any vertex in E .

Reference Scenario. For purpose of illustration, we describe a running example that we adopt throughout this paper (also further elaborated in Sect. 5). Consider the integration of a version-control and online source-code repository system *Github*, with a story-tracking system *Pivotal Tracker*. Often code traceability, collision and bug-repair require effective peer review and collaboration. The integration of these two tools would provide a powerful combination.

Figure 1 illustrates a canonical graph of possible relationships between entities, (where entities are structured data-objects with unique identifiers, and instance of entity-types, i.e. *APIs*, *Operations*, *Resources*, *Events* and *Tasks*). Illustrating the above scenario, we may learn that Pivotal Tracker consists of a few *Operations* that manipulates Story and a *StoryFinished* Event that’s

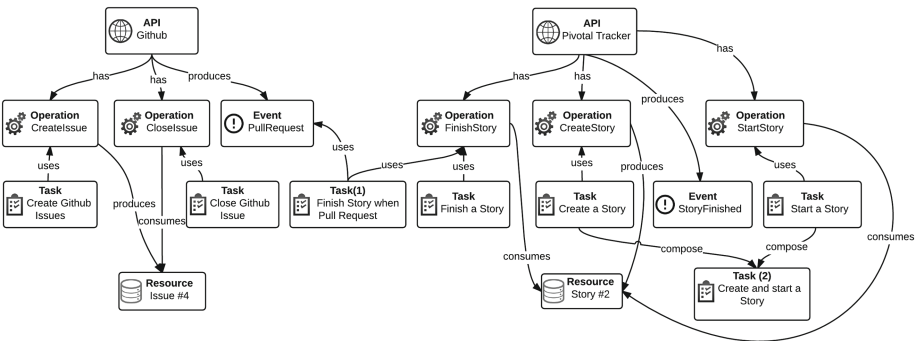


Fig. 1. A canonical graph of possible relationships between entities in the Knowledge Graph for case management systems.

¹ <http://www.google.com/insidesearch/features/search/knowledge.html>.

produced when a Story is finished. In addition, the *Automated Task(1)* will trigger the *FinishStory Operation* when a *PullRequest* Event from Github is received, providing a simplified work environment for the developer. In Pivotal Tracker, after a story is created, it has to be started manually afterwards to initiate the story. To simplify these steps, *Task (2)* can be used to shortcut this procedure. In the following, we define specific *entity-types* that are relevant:

API. Application Program Interface, represent the plethora of tools (e.g. storage, location, social-networking, etc.) exposed via Web-service interfaces (e.g. REST, WSDL). An API thereby encapsulates the *endpoint* and *operations*, (and other relevant information, e.g. OAuth protocol, refer [11]). We build upon our previous work [4], for a unified-structured approach of Web-APIs in the KB.

Operation represent the set of operations offered by a specific service (e.g. *CreateIssue* on Github, or *CreateStory* on Pivotal Tracker). Operations may *consume* or *produce* resource-types; and may also be used as *actions* within *tasks*.

Resource represent the input or output for an API. In this context, resources may have various granularities, from a large dataset to an entity representing issues in Github, stories in Pivotal Tracker, and even pdf files in Dropbox. To proficiently represent resources, we utilize JSON-Schema Draft v4.

Event is the record of an activity. Denoted E and consisting of attributes $\{T, R, \tau, D\}$, where T is the type of the event (e.g. Github provides 25 different event², such as *Push*, *Issue*, and *Fork*); R is the actor (i.e. person or device); τ is the timestamp; and D is a set of data elements recorded with the event (e.g. the task associated with the event). Events may be fine-grained (e.g. concrete events defined at the API level), or coarse-grained that capture a pattern relevant to a collection of resources (e.g., related stories in Pivotal Tracker).

Task represents the set of operations to achieve a defined goal, (ranging from a single to a combination of several API endpoints). Moreover, we propose the novel feature of “*context*”, which describes the “*intent*” and “*objective*”. This is especially useful to human process-designers or human-driven search engines. In this manner, operation endpoints may be mixed-and-matches between different APIs, to provide a more comprehensive *Task* that precisely targets a particular use-case. For example, the task *CreateIssue* may conform to different intents such as *report bugs*, *create pull requests* or *request feature*. Functionally, there are two *task-types*: *Automated* and *Manual*. The former assigned to an *Event* that may trigger the task, while the latter only triggered manually by an actor.

Tasks may also have sub-tasks, in order to help reduce complexity. For example, a *Code Commit* Task could be defined as a set of sub-tasks containing *GithubPush* and *GithubPullRequest* APIs. Utilizing Resources and Tasks nodes, we can identify potential service integration patterns, i.e.: If a Task T_A produces Resource R and Task T_B consumes Resource R , then Task T_A and T_B can be invoked in a sequence. For example, the Task *CreateIssue* produces an *Issue* and Task *EditIssue* consumes an *Issue*, therefore we can state that *EditIssue* can be invoked after *CreateIssue* on the same resource.

² <https://developer.github.com/v3/activity/events/types/>.

Relationship is denoted as $R = (E_1, E_2)$, which indicates a connection between entities E_1 and E_2 . As illustrated, we have seven types of relationships: “API has OPERATION”, “TASK use OPERATION”, “TASK use EVENT”, “OPERATION consumes RESOURCE”, “OPERATION produces RESOURCE”, “TASK compose TASK”, “OPERATION trigger EVENT”. In addition, there is “EVENT compose EVENT” indicating that an event is complex, and is representative of some pattern composed of several other events.

3 Knowledge-Reuse-Driven and Declarative Case Definition Language

The notion of *Case* conceptualises a lightweight process (set of service interactions). A case is thus defined to consist of: *Tasks*, *People* and *Event*. *Tasks* indicate the services and the features needed in the interaction. *People* describes who have access to the Case, especially the *owner* who has the privilege to edit the case. Cases can contain both automated tasks. As well as manual tasks, when human discretion and thereby intervention is required. However, *Tasks* can also monitor *Events* (or patterns thereof) which may serve as notification to participants (e.g. perform some manual task). Moreover, as *Cases* themselves are represented as nodes which can be curated and reused in a modular manner. While the platform is exposed via a RESTful interface, we further propose a higher-level command-line Case Search, Definition and Interaction Language.

3.1 Knowledge-Reuse Language

Selecting the required tasks for a Case may not be trivial with an extensively populated knowledge-graph. We thus propose an effective search component (utilizing the index of the tasks objective and an iterative keyword search approach [10]). The closest matching tasks are thus recommended to the user, based on the objective tags described in Sect. 2. Albeit, the decision of whether a Task (or Sub-task) should be included finally relies on the Case designers’ discretion (Fig. 2).

```

expression ::= "<op><keywords>"
op         ::= "task"      #matches tasks against all possible related keywords
           | "resource"   #tasks that are related to the resources matching the keywords
           | "input"      #tasks that consumes the resources matching the keywords
           | "output"     #tasks that produces the resources matching the keywords
           | "API"        #tasks that are directly related to the API specified
           | "event"      #tasks that are monitoring the events specified
           | "case"       #directly matches cases against all possible related keywords
keywords   ::= {<string>} #set of keywords to perform the search

```

Fig. 2. Search Language Syntax

```

expression ::= <new_case> | <extend>
new_case ::= "CREATE CASE" <name><own>
             [<shared>][<using service>][<monitor events>][<include tasks>]
extend ::= "EXTEND CASE" <name><new name><own>
            [<shared>] [<using service>][<monitor events>][<include tasks>]
own ::= "OWNED BY" < user >
shared ::= "SHARED WITH" < user > [{"", " < user >}]
using_service ::= "USING SERVICE" <service> [{"", " <service>}]
monitor_events ::= "MONITOR EVENTS" < event > [{"", " < event >}]
include_tasks ::= "INCLUDE TASKS" < task > [{"", " < task >}]

```

Fig. 3. Case Definition Language

3.2 Declarative Case Definition Language

Cases can be defined (or extended) using the language as defined below. The syntax contains governance policies over both people and services. For people, constructs such as OWNED BY (permission for editing/updating); SHARED WITH (permission for interacting). For services, the USING SERVICE construct indicates authorizations information - as the owner has to specify which authorization to share between the users of the Case. The MONITOR EVENTS construct details which events are to be monitored and notified to the participant users. While INCLUDE TASKS configures tasks related to the interaction (Fig. 3).

Figure 4 illustrates an example of a *GitHub Code Review Process with Pivotal Tracker*. Ordinarily, using Github alone, the Lead Developer may review code

```

CREATE CASE "CodeReview"
OWNED BY "Project Manager"
SHARE WITH "Project Manager", "Lead Developer"
USING SERVICE "GitHub", "Pivotal Tracker"
MONITOR EVENTS PullRequest, #pull request has been 'received'
                  PRMerged, #pull request has been 'merged'
                  ReviewFinished #review has been completed
INCLUDE ACTION MergePR, #merge a pull-request on GitHub
                  CreateStory, #create a story on pivotal tracker
                  FinishStory, #mark a story finished on pivotal tracker
                  StartStory, #start a story on pivotal tracker
                  DeliverStory, #deliver a story on pivotal tracker
                  AcceptStory #mark a story as accepted on pivotal tracker
INCLUDE TASKS CreateReviewOnPR, #invokes 'CreateStory' to create a new story to do review
                  DeliverOnPR, #invokes 'DeliverStory' when a pull-request is received
                  MergeOnFinish #invokes 'MergePR' when a review story is done

```

Fig. 4. CodeReview Case Definition Example

by requiring the **Engineers** to submit their code in forms of Pull Requests³ and then review it on Github. However, if the project manager wish to monitor the review progress on Pivotal Tracker, they will have to manually create review tasks (stories) for lead engineer every time a review is needed. Using *CaseWalls*, we can simplify this process by defining a case and linking automated tasks to auto-create stories when receiving a pull request. Events can also be monitored, notification posted to interested participants and thereby manual tasks (e.g. upon completing the review, merge the code into upstream, and close review process) can also be accomplished.

3.3 Declarative Case Manipulation Language

Finally, we propose a interaction language to interact with the Case during execution. Interactions may be with both manual tasks (awaiting human intervention), as well as automated tasks (as in tapping into some 3rd-party service) (Fig. 5).

```

expression ::= <op> <params>
op         ::= < op task > | < op resource >
op_task   ::= "#" <task >
op_resource ::= "@" <resource>
params    ::= <resource> | {<input> "=" <value>}

```

Fig. 5. Case Manipulation Language

Figure 6 illustrates *CaseWalls* for 3 participants in the *Code Review* process defined earlier. On the left, we see a set of notifications - this informs the participants what actions to take (if any). Actions might be interacting with some external software service (e.g. [P] Pivotal Tracker, and [G] GitHub)), or performing some manual task (i.e. [MT]). However, moreover behind the scenes automated tasks (i.e. [AT]) are also being performed as defined in the Case. It is thus apparent without *CaseWalls* all interactions would be done manually, with little or no flexibility. Not only do *CaseWalls* help automate certain tasks, it also automates the notification process - thus making it more simpler for participants to identify what needs to be done. Subsequently, the interaction language can be used to call upon manual tasks in a simple manner. As for implementation, the semantics of the language are translated into Rule-based expressions for the purpose of execution (refer to next section, at Sect. 4.4).

³ <https://help.github.com/articles/using-pull-requests/>.

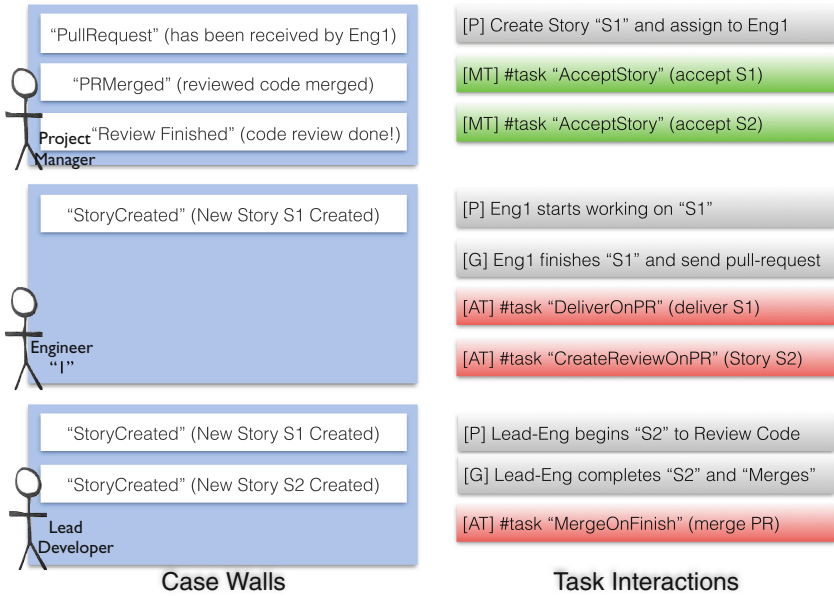


Fig. 6. CaseWalls with Illustration of Interactive Behavior

4 Implementation

4.1 Architecture

Figure 7 illustrates the system architecture and interaction of the main components of the *CaseWalls* platform. At the heart of the system is the *Knowledge-Graph (KG)* for Case-processes. It maintains the ontological relationships between key entities and facilitates task/case-based processes. Manipulation as well as effective searching of the case-based KG are conducted via the respective components (as shown). The *Event-Management* system: collects raw event-data from different services; and thereby, processes them using the patterns in the KG. We leverage our previous work for this, [4, 5]. This feeds into the *Rule-Engine* which performs pattern-matching and can infer which actions to perform by calling upon the *Task Execution Engine*.

Overall, *CaseWalls* has been implemented and exposed via a RESTful interface, together with an event-notification system. Event-notifications can either be *puSHed* or *Polled*, and effectively formulate the *Case-based Activity Walls*. Process participants can then take actions to interact/manipulate tasks. While at the time of writing the interface provided is programmatic only, future plans are to implement a GUI of the case-walls; or AS IS, we expect the platform to be extensible enabling 3rd-party higher-level and customizable applications.

4.2 Knowledge Graph

The knowledge graph plays a crucial part in the system. It needs to be robust and has to support complex graph query. We use *Neo4j* as the backend DB and

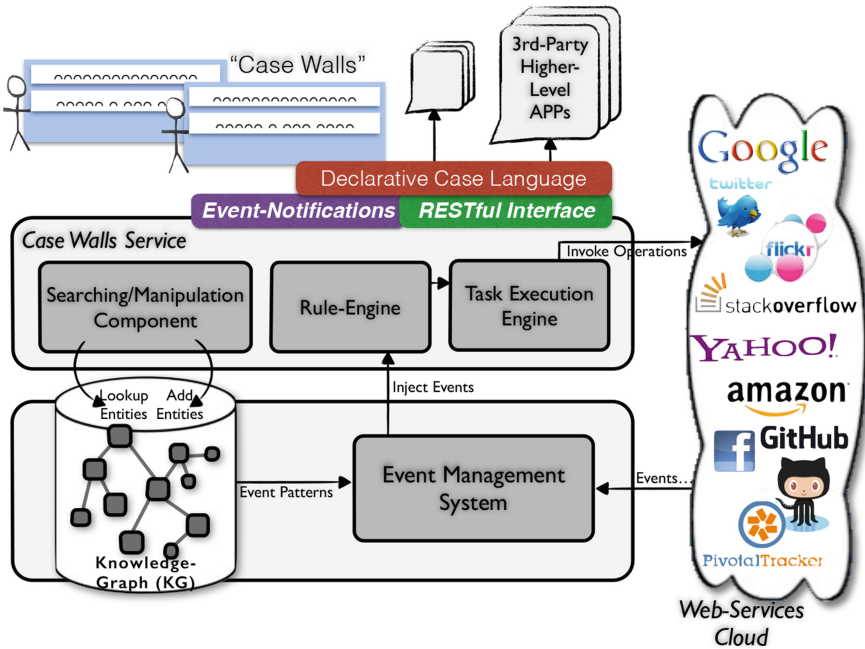


Fig. 7. System architecture of CaseWalls

build a typed graph database with a REST interface on top of it. Neo4j comes with the label system for its entities, which lets the user marks entities with a label but it doesn't enforce any schema except for unique constraint. Therefore we utilized JSON-Schema in the Knowledge graph for validating the JSON data.

By using Neo4j we can use its powerful graph query language call Cypher Query Language to query complex relationships. We can easily find a path from one Task to another. With this capability, we can provide the information about the interoperability between Tasks. We implement a RESTful API to enrich KG itself. Further details of the API can be found via swagger docs⁴. Figure 8 below shows the definition of the *DeliverOnPR* task curated in the knowledge-base.

4.3 Event Management System

The Event Management system is implemented to aggregate and process the events from different services. We use Fluentd⁵ for aggregating and dispatching the events received from various services, Norikra⁶ and Esper EPL⁷ for processing and generating high level events. For instance, defining an event with only

⁴ <https://raw.githubusercontent.com/freehaha/case-wall-api/master/case.yaml>.

⁵ <http://www.fluentd.org/>.

⁶ <http://norikra.github.io>.

⁷ <http://esper.codehaus.org>.

```

Task DeliverOnPR:
{
  "id": "ad6c7cf6-d18b-4323-8bc8-9e56055c313a",
  "name": "DeliverOnPR",
  "description": "deliver a story when a pull request is received",
  "mapping": "{ \"current_state\": \"#delivered\",
    \"storyId\": \"$.storyId\", \"projectId\": \"$.projectId\" }",
  "type": "auto",
  "intent": [
    "pullrequest",
    "story",
    "deliver"
  ],
  "_type": "Task",
  "event": [
    "pull_request"
  ],
  "tasks": [
    "UpdateStory"
  ],
  "_created": 1432520794
}

```

Fig. 8. Task definition expressed in JSON for *DeliverOnPR*

the attributes we need (thus masking the different payload and/or structure of the events). Higher-level events also enable defining an event based on a series of targeted events rather than just a single event. Likewise, we utilize MongoDB⁸ and ElasticSearch⁹ for archiving and indexing event data. For event collection, we implement an event-collecting application that connect to different services. Collected events are then sent to Fluentd to dispatch and process. Since the majority of services today utilize OAuth authorization, we have implemented and allow user to authorize our application to collect events automatically for them. Two kinds of events collecting mechanisms are implemented:

1. **Pushing.** Services like SendGrid, Twilio, Google Drive, Pivotal Tracker let user register a callback url, often called webhook, where the service will send a request to when events arrive. There is also a protocol call *PupSubHubBub* proposed by Google trying to standardize this event delivery method.
2. **Polling.** Some services use the legacy event polling model, requiring the client to constantly check whether there are new events. Another variation of it developed lately is long polling, sometimes called comet, designed to reduce the connection overhead by establish a keep-alive HTTP request. Services providing this kind of mechanisms include Twitter, Twitch.tv and Plurk.

4.4 Orchestration Engine: Generating Rules

CaseWalls further layers a more higher-level and declarative case manipulation language (refer Sect. 3). To implement this language, the semantics are

⁸ <http://www.mongodb.org>.

⁹ <http://www.elasticsearch.org>.

translated into Rule-based expressions, denoted: $R_{type} : (Events \rightarrow Actions)$. This effectively means, rules are generated from case definitions. Once deployed, the event-bus can detect relevant event-patterns, and working in conjunction with the rules-engine provision the execution of cases. Since we mentioned there are two main types of tasks: *manual* and *automated*. There are also two corresponding types of rules. Automated-task rules, denoted $R_{automated}$ consist of service-related events (e.g. *PullRequest*); and task-actions (e.g. *DeliverOnPR*). Manual-task rules, denoted R_{manual} may additionally consist of special internal events and actions. While not always necessarily utilized, they are at the disposal of the developer in order to grasp better control over manual tasks. In particular, they may prove useful to manage UI components¹⁰ associated with manual tasks. For example, if an event e_i triggers some manual task to be performed, the rule $Rule_{manual} : (e_i \rightarrow a_x)$ may be defined, where a_x can prepare or perform some pre-processing to some UI component. Likewise, another rule $Rule_{manual} : (e_i.e_j \rightarrow a_y)$ could denote that the manual task has been completed, where e_j is some UI event that the task was completed, and a_y could then be some post-processing action.

To better demonstrate case orchestration rules in the case of automated-tasks, we illustrate as shown in Fig. 9 the series of rules (i.e. *DeliverOnPR*, and *CreateReviewOnPR*) that would generated using the example *Code Review* case that we defined earlier (refer Sect. 3).

```

"rules": [
  {
    "action": [
      "UpdateStory"
    ],
    "event": "pull_request",
    "map": {
      "projectId": "$.projectId",
      "storyId": "$.storyId",
      "current_state": "#delivered"
    }
  },
  {
    "action": [
      "CreateStory"
    ],
    "event": "pull_request",
    "map": {
      "projectId": "$.projectId",
      "name": "#review PR #{number}"
    }
  }
]

```

Fig. 9. Example of generated case orchestration rules

5 Evaluation

Evaluation Objectives. To evaluate overall effectiveness, we assessed the following hypotheses: *Case Walls* is capable of (a) Improving the **productivity** to *model*, *reuse* and *execute* customized service-oriented processes; and (b) Increasing the **efficiency** of *application maintainability* for agile service integration.

¹⁰ Even with some 3rd-party tools, developers may tap into the tool (e.g. via Webhooks as in the case of Github). Alternatively, we may also refer here to custom UI components built by the developer to reflect certain manual tasks.

Experimental Setup. To assess the validity, the experiment was conducted by implementing a real-life *use-case scenario*. Analysis was then conducted via *comparison* to other approaches (incl. Javascript, Java, BPEL, Yahoo! Pipes). We divided our scenario into 2 phases; where the latter phase was to add onto the former, thus assessing ease of *maintainability*. Overall productivity was then measured as: (a) Time taken to complete task; (b) Total number of lines-of-code (LOC) excluding white-space; and (c) Number of extra dependencies needed.

Use-Case Scenario (*Code Review and Development Cycle*). Version Control Systems (VCS) are very common in software engineering - they help avoid collision and improve traceability. While it is important to find where the bug is introduced and revert it, peer review also helps to bring forward discovery of such bugs. *Github* is one of the most popular online open-source repositories for code. Likewise, *Pivotal Tracker (PT)* offers a good story-tracking system, to help the team keep track of their progress. **Phase 1** of this scenario involves integration of these two tools in the basic workflow described below:

1. Project Manager PM *creates* a Story and assigned to Engineer.
2. Engineer *starts* working on the Story.
3. Engineer completes programming task and pushes onto Github.
4. Engineer *finishes* and *delivers* the Story.
5. PM *accepts/rejects* the delivery.

Effectively, Github + PT integration may be implemented by parsing commit messages for syntax in the form of: “**#(number)**”, such as: [*Starts #12345, #23456*] ... [*Finishes #12345*] ... [*Delivers #12345*]. If any such messages are detected, the corresponding action will be performed in PT. For example, if the engineer commit message containing [*Finishes #12345*], when Github receives this commit, it will automatically *finish* that story in PT. This helps simplify the workflow by eliminating the otherwise manual work done within PT.

While this basic integration provides an initial improvement to eliminate the manual *creation, start, finish* and *delivery* of a PT “story”, **Phase 2** involves adapting it to “*continuous integration (CI)*”. The notion of CI, as prominent in software engineering today, calls for “continuous” testing whenever new changes are made. This would thus significantly alter the semantics of the *deliver* action. This means, at *Step 4*, we may want to introduce additional (and iterative) stories (cf. Steps 2–4) for: *testing* and *deployment* before closing this change.

Experimental Results. We set out to prove (or disprove) our hypotheses; the results are illustrated in Fig. 10. Outrightly, BPEL was excluded as real-time events are not available without writing custom extension to the engine. The same applied for Yahoo! Pipes, as it could not receive realtime events via webhooks¹¹. Hypothesis H(a) was evaluated as the time to complete both tasks (excluding setup time) and LOC. Using *CaseWalls* this resulted in only 30 mins,

¹¹ A possible solution could be done using feeds (rss/atom) for receiving the events, and a web-query framework, such as YQL to make requests. However, doing so is less interactive, less efficient and also requires writing sufficiently complex javascript.

compared to an average of 245 mins using other approaches (decrease of $\sim 88\%$); while LOC was 53 compared, to an average of 376 (decrease of $\sim 86\%$), respectively. For *CaseWalls* setup included time to create all the Operations/Events/Tasks in the KG; whereas for Javascript/Java this meant downloading and installing the requisite SDKs (where applicable). *CaseWalls* also resulted in less overall setup time. H(b) was then measured as the cost of implementing *Phase 2* (including any setup time). *CaseWalls* resulted in only 25 mins to implement compared to an average of 220 mins (decrease of $\sim 89\%$), with 30 and 93 LOC respectively (decrease of $\sim 67\%$).

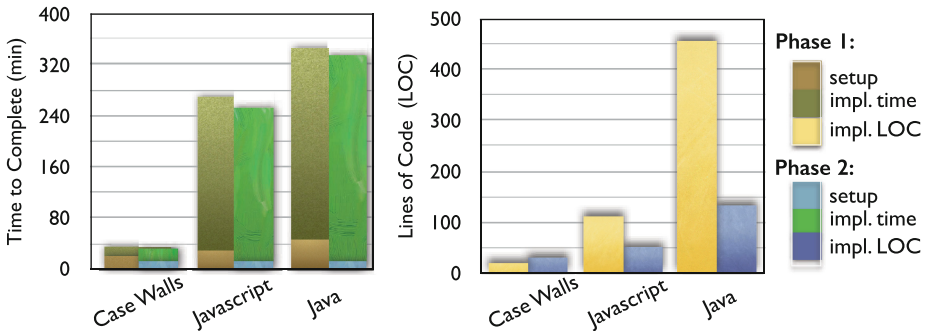


Fig. 10. Evaluation results for GDrive contribution calculator use-case

Overall, it was clear both *time* and *LOC* is significantly reduced when using *CaseWalls*. We thus validate both our hypotheses as true with very promising results. Moreover, our approach did not require any additional libraries, whereas others required on average at least 2–3. *CaseWalls* also provided the facility of increased transparency, as well as agile participant control - compared to other solutions which were rather rigid. In light of these results, this evaluation study successfully demonstrates the anticipated benefit of our proposed approach.

6 Related Work and Concluding Remarks

The ubiquitous access to thousands of APIs offer tremendous potential in modern App development. For example, ProgrammableWeb records some 13,495 APIs over numerous categories, including: financial, mapping, social-networking, etc. The inevitable key to success is thus ‘API integration’ - as the empowerment to compose disparate services (rather than reimplementing) will reap great reward. Currently, there are a plethora of SaaS-enabled tools that aim to fulfill a specific user-need, however these ready-made solutions often imply conforming to a fixed set of embedded features with little room for user-customization. On the other side of the spectrum, BPM sought to offer customizable process-support over disparate services, albeit it suffered significantly from a lack of flexibility.

In the following, we analyze these two technological polarities. We then offer an innovative set of guiding principles for converging these polar extremes - which is the refreshing outlook we have adopted in positioning our work in this paper.

Web-Services/API Integration Development. Modern service-oriented systems aim to support services integration, [3, 8, 12]. For instance, the Enterprise Service-Bus (ESB) was an early and still prevalent method for handling message-exchange over heterogenous and distributed components. *Apache ServiceMix* [14], is one example providing advanced features. Micro-services are yet another alternative approach that are well aligned with cloud provisioned services, and tools such as *Netflix Asagard* [15], and *Pivotal Cloud Foundry* [13], have emerged. Albeit, these methods still do not alleviate even professional programmers from being coerced in understanding the various low-level service APIs, as well as working directly with procedural programming constructs to create and maintain complex applications. This leads to an inflexible and costly environment which adds considerable complexity, demands extensive programming effort, multiple and continuous patches, and perpetual solutions, [12].

Process-Oriented Service Programming. Advanced process support systems (e.g. BPEL) and Mashup environments (e.g. Yahoo! Pipes) aimed to counteract the above challenges, while also appealing to the less-technical. However, while they helped avoid low-level API programming, composition environments significantly lacked the productivity support tools that developers were used to whilst programmers using IDEs, (e.g. code search and discovery, ease of reuse, debugging, code generation), [12]. Moreover, they suffered from the lack of flexibility and cannot support run-time changes, [6]. This worsens as the variety of services and variations of application requirements and constraints increase, [4].

Case Management. Flexibility is imperative to transition composition systems from the realm of static and small-scale environments to that of large-scale computing, relying on highly unpredictable and evolving environments. Case-Management is an emerging step in the right directions, given many processes are knowledge-intensive and thereby human-driven, [1, 2, 9]. For example, a customer initiating a request for some services, the set of interactions among people, e.g. customer and relevant participants, and artifacts from initiation to completion is known as the ‘case’. However, while well conceptualized, much of its actual implementation remains vague and depends on its context of use, [7].

Summary. Reflecting on the above, we have discovered Web-services mirroring (at least conceptually), the evolution of database management systems (DBMS)s over the last 30-years. In effect, DBMSs have called for generic abstractions and declarative techniques (e.g., data-models, relational algebras, declarative query techniques) for simplifying the design of complex applications and enabling high level manipulation of data. Similarly, we propose the following set of guiding principles that Web-service APIs should adopt: (i) Modularity, similar building blocks in terms of simple and useful models; (ii) Declarative analysis, including support for high-level language manipulation, integration and transformation; and (iii) Knowledge-preserving, such that API-related programming knowledge can be curated for future communal reuse.

Accordingly, *Case Walls* is a refreshing step towards this innovative direction, providing a framework to simplify the integration of disparate services and effectively build flexible customizable processes. Our knowledge-driven approach builds upon our previous work [6], and is inspired by efforts in general knowledge-graphs such as “linked data”. We have thus proposed a novel *Case Knowledge Graph (CKG)* to facilitate the organization, integration, querying, and reusing of the case management knowledge. Moreover, the ability for case-works to “re-use” process knowledge is a vibrant change to most existing process platforms.

Empowered by this knowledge graph, we also provided a novel case customization and deployment platform. And to even further increase user efficiency, we introduced a simple, declarative yet powerful language to query and analyze the knowledge graph. Unlike any previous works, *Case Walls* focusses on the transparency aspect of mid-process knowledge. The concept of “walls” thus act as an activity wall akin to social status updates, albeit instead updates are sourced as relevant events from case tasks. Participants are then empowered to track the case execution, and react/interact accordingly.

Experimental results shows promising results, in particular addressing the dimensions of increased user-efficiency. In future, we are excited to enhance and extend the language power and expressivity - as well as implement a novel graphical user-interface that mimics social-networking platforms. Whereby case-based process functionality can effectively be combined within everyday tasks. We are therefore very optimistic this work provides the foundation for future growth into a new breed of enhanced process-support.

References

1. Van der Aalst, W.M., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. *Data Knowl. Eng.* **53**(2), 129–162 (2005)
2. Swenson, K., et al.: Taming the Unpredictable Real World Adaptive Case Management: Case Studies and Practical Guidance. Future Strategies Inc. (2011)
3. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: *Web Services: Concepts Architectures and Applications*. Springer Publishing Company Incorporated, Heidelberg (2010)
4. Barukh, M.C., Benatallah, B.: ServiceBase: a programming knowledge-base for service oriented development. In: Feng, L., Bressan, S., Winiwarter, W., Song, W., Meng, W. (eds.) *DASFAA 2013, Part II*. LNCS, vol. 7826, pp. 123–138. Springer, Heidelberg (2013)
5. Barukh, M.C., Benatallah, B.: A toolkit for simplified web-services programming. In: Lin, X., Manolopoulos, Y., Srivastava, D., Huang, G. (eds.) *WISE 2013, Part II*. LNCS, vol. 8181, pp. 515–518. Springer, Heidelberg (2013)
6. Barukh, M.C., Benatallah, B.: *ProcessBase: a hybrid process management platform*. In: Franch, X., Bhiri, S., Ghose, A.K., Lewis, G.A. (eds.) *ICSOC 2014*. LNCS, vol. 8831, pp. 16–31. Springer, Heidelberg (2014)
7. Böhringer, M.: Emergent case management for ad-hoc processes: a solution based on microblogging and activity streams. In: Muehlen, M., Su, J. (eds.) *BPM 2010 Workshops*. LNBIP, vol. 66, pp. 384–395. Springer, Heidelberg (2011)

8. Geambasu, R., Cheung, C., Moshchuk, A., Gribble, S.D., Levy, H.M.: Organizing and sharing distributed personal web-service data. In: Proceedings of the 17th International Conference on World Wide Web, pp. 755–764. ACM (2008)
9. Kaan, K., Reijers, H.A., van der Molen, P.: Introducing case management: opening workflow management’s black box. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 358–367. Springer, Heidelberg (2006)
10. Klemisch, K., Weber, I., Benatallah, B.: Context-aware UI component reuse. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 68–83. Springer, Heidelberg (2013)
11. Nolan, D., Lang, D.T.: Authentication for web services via OAuth. In: Nolan, D., Lang, D.T. (eds.) XML and Web Technologies for Data Sciences with R, pp. 441–461. Springer, New York (2014)
12. Pautasso, C., Zimmermann, O., Leymann, F.: Restful web services vs. big web services: making the right architectural decision. In: Proceedings of the 17th International Conference on World Wide Web, pp. 805–814. ACM (2008)
13. Pivotal-Cloud-Foundry. <http://pivotal.io/platform-as-a-service/pivotal-cloud-foundry>
14. ServiceMix, A.: Apache servicemix 3. x users’ guide. Apache ServiceMix Community (2007). <http://incubator.apache.org/servicemix/users-guide.html>. (Cited on p. 72, 73 and 149)
15. Sondow, J.: Asagard: web-based cloud management and deployment. The Netflix Tech Blog (2012)

Business Process Management

Correlation Mining: Mining Process Orchestrations Without Case Identifiers

Shaya Pourmirza^(✉), Remco Dijkman, and Paul Grefen

Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{s.pourmirza,r.m.dijkman,p.w.p.j.grefen}@tue.nl

Abstract. Process discovery algorithms aim to capture process orchestration models from event logs. These algorithms have been designed for logs in which events that belong to the same case are related to each other - and to that case - by means of a unique case identifier. However, in service oriented systems these case identifiers are usually not stored beyond request-response pairs, which makes it hard to relate events that belong to the same case. This is known as the *correlation challenge*. This paper addresses the correlation challenge by introducing a new process discovery algorithm, called the *correlation miner*, that facilitates process discovery when events are not associated with a case identifier. Experiments performed on both synthetic and real-world event logs show the applicability of the correlation miner.

Keywords: Process mining · Process discovery · Event correlation

1 Introduction

Over the past decade, there has been an increasing interest in the area of process mining [2, 10, 16–18, 20–22]. The goal of process mining is to extract information about processes from event logs, i.e., execution histories. One of the prominent branches of process mining is process discovery [20], which concerns itself with generating an orchestration model from an event log.

Process discovery techniques assume that an event log contains at least, for each recorded event: (i) a reference to the executed activity, (ii) a reference to the case for which the activity was executed, and (iii) the timestamp at which the activity was completed [21]. Table 1 shows an example of the event log involving 30 events in 10 cases. The main idea behind discovery algorithms is to merge, cluster and aggregate the different cases in an event log and generate a suitable orchestration model based on that. For example, considering the event log from Table 1, it is possible to capture three different traces of execution: $\langle A, B, E \rangle$ for Cases 1, 4 and 5; $\langle A, D, E \rangle$ for Cases 2, 3, 8 and 9; and finally, $\langle A, C, E \rangle$ for Cases 6, 7 and 10. Figure 1 presents the corresponding orchestration model, as it would have been mined by the Disco process mining tool [11].

The research leading to these results has received funding from the European Union's Seventh Framework Programme under grant agreement 2012-318275 (GET Service).

Table 1. An example event log

Case	Activity	Timestamp	Case	Activity	Timestamp	Case	Activity	Timestamp
1	A	00:20	4	B	05:04	7	E	09:17
1	B	02:04	4	E	07:26	8	A	06:20
1	E	02:32	5	A	03:40	8	D	08:36
2	A	02:15	5	B	05:59	8	E	10:03
2	D	03:14	5	E	07:49	9	A	06:41
2	E	05:06	6	A	04:18	9	D	08:56
3	A	02:27	6	C	07:08	9	E	10:20
3	D	04:17	6	E	09:05	10	A	07:13
3	E	06:51	7	A	05:54	10	C	09:10
4	A	03:06	7	C	07:30	10	E	10:26

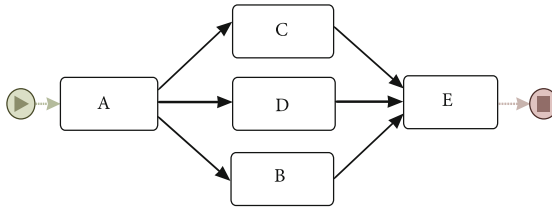


Fig. 1. Orchestration model generated from the example event log

While process mining techniques require an event to be associated with a case, making this association can be problematic. This is also referred to as the *correlation challenge* [18]. This challenge arises, for example, when information about the occurrence of activities is stored in separate service oriented systems and there is no obvious identifier to correlate the occurrences. It is considered especially problematic in service mining [19], because correlations between messages that are being exchanged for the same case, may not be stored beyond request-response pairs. When associations between events that belong to the same case are not present, process discovery becomes impossible, as illustrated by Table 2, which shows the event log from Table 1 without case identifiers. In this event log it is not possible to identify cases and, consequently, it is not possible to cluster and aggregate them into an orchestration model.

Therefore, the goal and contribution of this paper is to develop a process discovery technique that can discover orchestrations when events are neither correlated by case identifiers nor by additional data elements. Although extensive research has been carried out on both process discovery and event correlation techniques, to the best of our knowledge, only one [8] such process discovery technique exists that will be further discussed in this paper.

The remainder of this paper is structured as follows. Section 2 presents some preliminaries regarding the event logs and orchestration models. Section 3 introduces the actual technique that has been developed to capture orchestration

Table 2. An example event log without case identifiers

Activity	Timestamp	Activity	Timestamp	Activity	Timestamp
A	00:20	B	05:04	C	07:30
B	02:04	E	05:06	E	07:49
A	02:15	A	05:54	D	08:36
A	02:27	B	05:59	D	08:56
E	02:32	A	06:20	E	09:05
A	03:06	A	06:41	C	09:10
D	03:14	E	06:51	E	09:17
A	03:40	C	07:08	E	10:03
D	04:17	A	07:13	E	10:20
A	04:18	E	07:26	E	10:26

models from event logs without case identifiers. Section 4 discusses the evaluation setup and summarizes the obtained results. Section 5 compares our technique with related literatures. Finally, Sect. 6 concludes the paper by giving a brief overview and its findings.

2 Preliminaries

In the literature, an event log has been defined as a multiset of cases [18]. However in this paper, we cannot reuse this definition since we have no explicit case. Therefore, we define an event log as follows.

Let A be a set of activities and T be a set of timestamps. $L \subseteq A \times T$ is an *event log*. We use E_a to denote a set of events referring to the activity $a \in A$, such that $E_a = \{(a, t) | (a, t) \in L\}$. In addition, we represent the timestamp at which event $e \in E_a$ has happened by t_e . Note that, strictly speaking, multiple events for the same activity can happen at the same time. However, for simplicity, this is not covered by our definition. While the occurrence of identical events is sufficiently rare to not create problems with the algorithm, it does require some pre-processing of the log to make sure that identical events are distinguished from each other (e.g. by adding a suffix to the timestamp).

There exist various notations in which the orchestration model that is the result of applying a process discovery algorithm can be represented (e.g. Petri-Nets and BPEL) [18]. In this paper, we adopt a rather abstract definition of an orchestration model, which is in line with the notation that is used by the Disco tool for process discovery [11].

The orchestration model is a directed *weighted* graph $G = (\mathcal{V}, \mathcal{E}, \omega)$ such that:

- $\mathcal{V} = \{(a, n) | a \in A \wedge n = |E_a|\}$, where A is the set of activities and n indicates for each node the number of occurrences of that activity in the event log;
- $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges;

- $\omega : \mathcal{E} \rightarrow \mathbb{N}$ maps edges to natural numbers, where $\omega(a, b) = n$ if and only if there are n observations of activity a being directly followed by activity b in some case in the log. We require that $e \in \mathcal{E}$ if and only if $\omega(e) > 0$.

For a node $v \in \mathcal{V}$, the in-degree, denoted $deg^{in}(v)$, is $\sum_{(w,v) \in \mathcal{E}} \omega(w, v)$. The out-degree, denoted $deg^{out}(v)$, is $\sum_{(v,w) \in \mathcal{E}} \omega(v, w)$. When $deg^{in}(v) = 0$, we call the node a source node. When $deg^{out}(v) = 0$, we call the node a sink node.

It is important to note that, for any node $(a, n) \in \mathcal{V}$ that is not a sink node, it must hold that $deg^{out}(a) = n$, because the number of cases that pass through the node must also continue to another node. Similarly, for any node $(a, n) \in \mathcal{V}$ that is not a source node, it must hold that $deg^{in}(a) = n$. We call these constraints the *orchestration graph rule*.

3 Correlation Mining Technique

The basic idea of the correlation miner is based on the orchestration graph rule, defined in the previous paragraph, which states that the number of cases that pass through an activity must be equal to the number of incoming and outgoing cases for that activity. We can count the number of cases that pass through an activity a by counting the number of occurrences $|E_a|$ of that activity in the log. Subsequently, we can draw the edges between the activities, in such a way that orchestration graph rule is met. This is a constraint programming problem that can be solved using integer linear programming. At this point we assume that the source and sink nodes are set manually.

As an example, Fig. 2a shows an orchestration model that can be constructed from the log in Table 2. In this log A occurs 10 times, while D occurs 4 times. We manually select A as a source node. Consequently, it has an in-degree of 0. The number of occurrences of A corresponds to the out-degree of A , which is 4 (to D) plus 6 (to E) equals 10. Similarly, the number of occurrences of D corresponds to the in-degree and the out-degree of D . In this way, creating an orchestration model that meets the orchestration graph rule for all activities, produces Fig. 2a. However, Fig. 2b, c, and d are also models that meet this rule. While Fig. 2c is the model that matches the original log from Table 1 best, the orchestration graph rule alone is not enough to determine that.

As the example shows, it is often possible to create more than one model that meets the criteria. Therefore, additional measurements from the event log are required in order to select the best model. To this end, the correlation miner creates two matrices, the *Precede/Succeed matrix* and the *Duration matrix*. Each score in Precede/Succeed matrix, $P/S_{i,j}$, indicates the fraction of events referring to activity i that have occurred before events referring to activity j . If this score is high, it is more likely that there is an edge from i to j . Each score in Duration matrix, $D_{i,j}$, indicates the average time or standard deviation (we will experiment with both alternatives) of the time difference between events referring to activity i and events referring to activity j . If this score is low, it is more likely that there is an edge from i to j . In the remainder of this section,

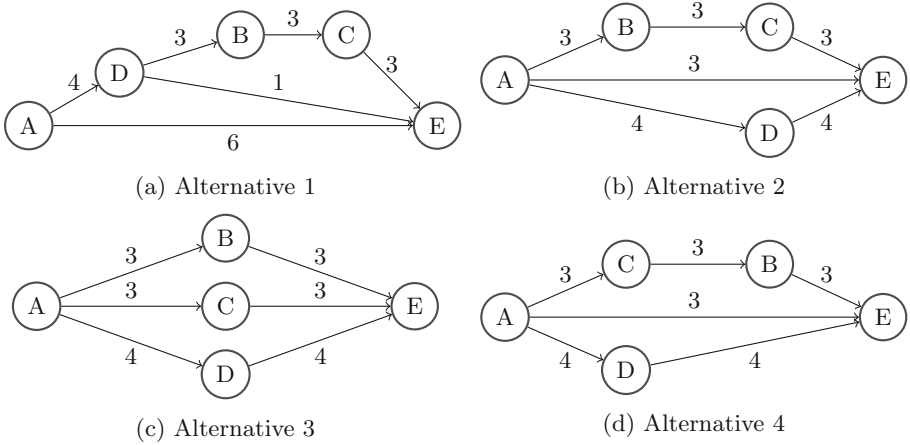


Fig. 2. Feasible graphs of example log

we explain how these matrices are computed, how they are used to construct the linear programming problem, and ultimately how the orchestration model is constructed.

3.1 Precede/Succeed Matrix

The first step in the correlation miner is to calculate the Precede/Succeed matrix. This matrix is a square matrix of order n , with $n = |A|$. Let i represent a row in the matrix as well as an activity from A and let j represent a column and an activity. A value $P/S_{i,j}$ in the matrix represent the fraction of events from i that occurred before events from j and is computed as follows. $\Omega_{i,j} = \{(e, f) | e \in E_i \wedge f \in E_j\}$ is the set of all pairs of occurrences of i and j . Furthermore, given events e and f , let $b(e, f)$ be a function that is 1 if $t_e < t_f$ and 0 otherwise.

$$P/S_{i,j} = \frac{\sum_{(e,f) \in \Omega_{i,j}} b(e, f)}{|\Omega_{i,j}|}$$

Returning to our example (Table 2), $P/S_{C,E}$ can be computed as follows. $\Omega_{C,E}$ contains 30 elements since $|E_C| = 3$ and $|E_E| = 10$. $\sum_{(e,f) \in \Omega_{C,E}} b(e, f) = 17$, because in 17 cases C occurred before E . Therefore, $P/S_{C,E} \approx 0.57$.

	A	B	C	D	E
A	0.00	0.47	0.97	0.73	0.88
B	0.53	0.00	1.00	0.67	0.90
C	0.03	0.00	0.00	0.33	0.57
D	0.26	0.33	0.67	0.00	0.70
E	0.12	0.10	0.43	0.30	0.00

Fig. 3. P/S matrix for example log

Similarly, we can calculate the value for each pair of activities in a given event log. Figure 3 illustrates the matrix that is calculated based on the example log.

For the performance reasons, we can remove some pairs to not include them for consideration during the next two steps of the algorithm. This can be done via a threshold filter. The goal of the threshold filter is to remove pairs of activities that have a very low probability of forming an edge. For example, in Fig. 3, we have $P/S_{A,C} = 0.97$ and $P/S_{C,A} = 0.03$. These values indicate that it is likely that there is an edge from A to C , but no edge from C to A . The challenge is to select a threshold that removes as many pairs as possible in order to increase performance, but to prevent false positives, because pairs that are removed at this stage will never become edges in the orchestration model.

3.2 Duration Matrix

The second step in the correlation miner is to generate the Duration matrix. This matrix is a square matrix of order n , with $n = |A|$. Let i represent a row in the matrix as well as an activity from A and let j represent a column and an activity. A value $D_{i,j}$ indicates the average time difference or the standard deviation of the time difference between events referring to activity i and events referring to activity j .

In order to calculate the value of $D_{i,j}$ we need to have a *mapping* between elements of E_i and elements of E_j . The idea behind this mapping is that it maps events that belong to the same case. We compute the mapping, by relating the events in such a way that the variance in the time difference is as low as possible, because we argue that the time difference between events from the same case have a more constant probability distribution than the time difference between events from different cases. This principle is not likely to yield a perfect mapping, in the sense that events are mapped if and only if they belong to the same case. However, a perfect mapping is not necessary. We just need a mapping that has a time distribution that is close enough to the perfect mapping.

Formally, let $\Omega_{i,j}$ be defined as in Sect. 3.1. Then we compute the mapping $M_{i,j} \subseteq \Omega_{i,j}$ that satisfies the following constraints.

$$\begin{aligned} \forall (e, f) \in M_{i,j} : \exists (e, f') \in M_{i,j} \Rightarrow f = f' \\ \forall (e, f) \in M_{i,j} : \exists (e', f) \in M_{i,j} \Rightarrow e = e' \\ \forall (e, f) \in M_{i,j} : t_e < t_f \end{aligned}$$

These constraints state that each event can be mapped at most once and that each event from E_i must be mapped to an event from E_j that occurs at a later point in time. In addition, the mapping must maximize its size (i.e. as many events must be mapped as possible), while minimizing the standard deviation of the time difference between mapped events. With this mapping, we can compute $D_{i,j}$ as either the mean $\overline{\Delta M_{i,j}}$ or the standard deviation $\sigma_{\Delta M_{i,j}}$.

$$\begin{aligned} |M_{i,j}| \text{ must be maximized} \\ \sigma_{\Delta M_{i,j}} \text{ must be minimized, where } \Delta M_{i,j} = \{t_f - t_e | (e, f) \in M_{i,j}\} \end{aligned}$$

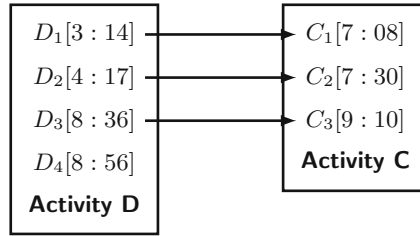


Fig. 4. Mapping of events E_D to events E_C

$$D = \begin{matrix} & A & B & C & D & E \\ \begin{matrix} A \\ B \\ D \\ E \end{matrix} & \begin{pmatrix} 0 & 74 & 109 & 108 & 220 \\ 54 & 0 & 214 & 164 & 69 \\ 6 & 0 & 0 & 87 & 42 \\ 0 & 106 & 150 & 0 & 124 \\ 0 & 102 & 62 & 96 & 0 \end{pmatrix} \end{matrix}$$

Fig. 5. Duration matrix for example log

Returning to our example (Table 2), $D_{D,C}$ can be computed as follows. Figure 4 shows the mapping from events E_D to events E_C that satisfies the constraints above. The average time difference between mapped events is approximately 153s. Similarly, we can calculate the value for each pair of activities in a given event log. Figure 5 illustrates the matrix that is calculated based on the example log. Note that the time difference in the matrix is slightly different from the time difference that we computed above. This is because we implemented an efficient (greedy) algorithm to compute the mapping, which does not always produce the mapping that satisfies all constraints. In particular, it does not always return the mapping with the lowest standard deviation, but rather the mapping with a local minimum. Although a lower score of $D_{i,j}$ indicates that it is more likely that there is an edge from i to j , a score of 0 means it is not possible to have such an edge, since we assume that events cannot happen at the same time in the event log.

3.3 Orchestration Model Construction

The final step in the correlation miner is to generate the orchestration model. We first explain how all possible orchestration models can be generated, taking the orchestration graph rule into account. Then we explain how the best orchestration model can be selected from all possible orchestration models, using the Precede/Succeed matrix (Sect. 3.1) and Duration matrix (Sect. 3.2).

In order to construct feasible models, we formulate our problem as an *Integer Linear Programming* (ILP) problem. Each possible edge (i, j) between activities becomes a variable x_{ij} of our ILP problem. The value that is assigned to each variable, x_{ij} , indicates the frequency with which an event for activity i is directly

followed by an event from activity j in a case. Therefore this value can be seen as an edge weight for the edge that connects the node that refers to activity i to the node that refers to activity j . Let \mathcal{X} denote the set that contains the introduced variables. Also, let A_s be the set of activities that we consider start activities and A_e be the set of activities that we consider end activities. We can reduce the problem space, by removing all variables:

- x_{ij} that represent unlikely edges, because of a $P/S_{i,j}$ or $D_{i,j}$ thresholds;
- $x_{is}, s \in A_s$, which represent incoming edges to start activities; and
- $x_{ei}, e \in A_e$, which represent outgoing edges from end activities.

Subsequently, for the resulting set of variables \mathcal{X} , we can formulate the constraints of our ILP problem as follows. The lower bound for each variable is 0, while the upper bound for each variable is $\min(|E_i|, |E_j|)$, because these are the minimum and the maximum number of cases that can flow on the edges. The sum of the number of cases on the incoming edges of an activity a , must be equal to the number of times an event E_a occurs for that activity. Similarly, the sum of the number of cases on the outgoing edges of an activity a , must be equal to the number of times an event E_a occurs for that activity. Consequently, the constraints become:

$$\begin{aligned} x_{ij} &\geq 0 && \text{for each } a \notin A_s : \sum_{x_{ia} \in \mathcal{X}} x_{ia} = |E_a| \\ x_{ij} &\leq \min(|E_i|, |E_j|) && \text{for each } a \notin A_e : \sum_{x_{ai} \in \mathcal{X}} x_{ai} = |E_a| \end{aligned}$$

Returning to the example of the log from Table 2, we can formulate the following constraints. $x_{AE} \geq 0$ and $x_{AE} \leq 10$. Similarly, $x_{AB} \geq 0$ and $x_{AB} \leq 3$. Also, $x_{AE} + x_{BE} + x_{CE} + x_{DE} = 10$ and $x_{AB} + x_{AC} + x_{AD} + x_{AE} = 10$. In addition, $x_{CB} + x_{CD} + x_{CE} = 3$, and $x_{AC} + x_{BC} + x_{DC} = 3$. Note that, since x_{CA} represents an incoming edge to a start node A and x_{EC} represents an outgoing edge from an end node E , these two variables have been removed and consequently are ignored in these constraints.

By applying these constraints, we can construct all feasible orchestration models. Now in the second part of this section, our goal is to select the model, from all feasible ones, that best represents the event log behavior. For this purpose, we employ matrices that we computed in the first two steps of the correlation miner algorithm in order to formulate an *objective function* to select the *optimal* model out of all feasible ones, using ILP principles.

The matrices from step 1 and step 2 provide some evidence to suggest that if activity pair (i, j) contains a high value for $P/S_{i,j}$ and a low value for $D_{i,j}$, these two activities may have a higher chance to form an edge. Based on this statement, our objective is to select the model that has a higher cumulative value of P/S s and a lower cumulative value of D s for the edges. To achieve this objective, we define a *coefficient* for each variable x_{ij} , in our ILP in order to consider these values. Since we are interested in a high value of $P/S_{i,j}$ and a low

value of $D_{i,j}$, we define an *edge ratio* as $\frac{D_{i,j}}{P/S_{i,j}}$ for each variable x_{ij} that must be minimized. Note that, we need to keep the sum of edge ratios for each edge independent of the assigned value of the referring variables and count the edge ratio if there is *an* edge, irrespective of the weight of that edge. Thus, ideally, each x_{ij} is divided by its value and set to 0 if $x_{ij} = 0$ in order to eliminate their influence. However, this is not possible in an ILP. Therefore, in order to reduce this influence we divide each variable by its upper bound. In conclusion, we formulate the objective function for our ILP as follow:

$$C_{ij} = \frac{D_{i,j}}{P/S_{i,j}} \cdot \frac{1}{\min(|E_i|, |E_j|)} \quad \text{minimize} \quad \sum_{x_{ij} \in \mathcal{X}} C_{ij} \cdot x_{ij}$$

Returning to the example log (Table 2), by considering the constraints, multiple models can be constructed as shown on Fig. 2. Now, we can calculate the objective function for each of these orchestration models and select the one that has a lowest value as an output of our algorithm. The value for the orchestration model from Fig. 2c is the lowest. Therefore, we select the orchestration model of Fig. 2c as the final result of our correlation miner.

The algorithm that we described above may return a model that contains cycles. However, since the correlation miner only deals with acyclic orchestrations, these cycles should be removed. Therefore, it employs Johnson’s algorithm [12] to capture all edges that are involved in cycles in the model. It then calculates the edge ratio for these edges and removes the edge with highest edge ratio. The correlation miner then reruns the third step of the algorithm and repeats this until a model without cycles is returned. As the main loop of the algorithm is ILP solving, the time complexity is polynomial.

4 Evaluation

This section presents the evaluation of the correlation miner. It first presents the setup of the evaluation in Sect. 4.1. Then it presents two evaluations, one using synthetic event logs (Sect. 4.2) and one using a modified version of a real-world event log (Sect. 4.3).

4.1 Evaluation Setup

In order to conduct our evaluation, we have implemented the correlation miner as a Java application. Since the algorithm uses the Gurobi ILP Solver [15], which is commercial software, one needs to install this software on his machine with a proper license. For the interested reader we provide on-line access to the algorithm¹, which can only be used for the academic purposes. Moreover, we are currently investigating the licensing issue in order to implement the algorithm as a ProM [6] plug-in.

¹ <http://is.ieis.tue.nl/research/correlation>.

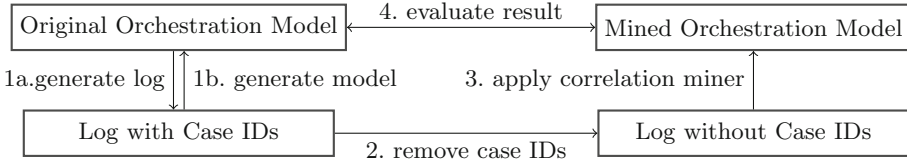


Fig. 6. Evaluation procedure

Figure 6 depicts the procedure for evaluating our algorithm. For the evaluation with synthetic logs, we first create synthetic orchestration models with particular properties, to investigate the effect that these properties have on our algorithm. This will be explained in detail in Sect. 4.2. Subsequently (step 1a), we generate synthetic logs for these models, using the BIMP simulator². For the evaluation with real-world logs, we take a log from practice. Subsequently (step 1b), we generate a orchestration model for that log. This will be explained in detail in Sect. 4.3. Now that we have both a log and a model, we remove the case identifiers from the log (step 2) to generate a log that can be used for correlation mining and (step 3) we apply the correlation miner. Finally (step 4), we assess the quality of the mined model.

In order to measure the quality of the mined models, we use *precision* and *recall*. Precision measures the fraction of edges in the mined model that are correct, i.e. are also in the original orchestration model. Recall measures the fraction of correct edges that have been found. These measures are defined as follows. Let TP be the set of edges that exist in the mined model and also in the original; FN be the set of edges that do not exist in the mined model but do exist in the original model; and FP be the set of edges that exist in the mined model but do not exist in the original model. Then precision and recall are defined as:

$$precision = \frac{TP}{TP + FP} \quad recall = \frac{TP}{TP + FN}$$

Returning to Fig. 2 and assuming that Fig. 2b is the mined model and Fig. 2c the original model, we can calculate precision and recall as follows. TP for these two models is 4, including AB , AD , CE and DE ; FN is 2, including AC and BE ; and finally FP is again 2, including BC and AE . Consequently, precision and recall for Fig. 2b are approximately 0.67.

In related work fitness and appropriateness have been introduced [16] as measures to evaluate the quality of a mined orchestration model. However, these measures evaluate the quality of an orchestration model as it is compared to a log. We evaluate the quality of the orchestration model as it is compared to another orchestration model; the orchestration model that should have been returned. By doing so, we can get more meaningful results. However, clearly

² <http://bimp.cs.ut.ee/>.

this is only possible when such an orchestration model indeed exists. In other situations fitness and appropriateness should be used.

4.2 Synthetic Event Logs

We evaluated the correlation miner using over 250 synthetic event logs with different properties. We varied properties that we assumed would have an effect on the quality of the mined orchestration model. Specifically, we generated logs:

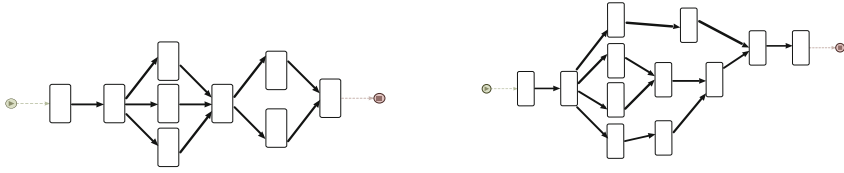
- using both structured and unstructured orchestration models [13] and using orchestration models with different numbers of branches, because these factors influence model complexity and we assume that more complex models are more difficult to mine;
- that contained different numbers of cases, because a higher number of cases provides more data to mine from and should therefore produce more accurate models;
- with different inter-arrival times and activity durations, because time properties play an important role in our algorithm and should therefore have an impact on the quality of the result.

In total we used 6 orchestration models to generate our logs. For reasons of space, we do not present all results, but rather the results for the model that produced the best results (Fig. 7a) and the model that produced the worst results (Fig. 7b). These models primarily vary with respect to whether they are structured or unstructured. The maximum number of branches does not vary much, because our evaluation showed that the number of branches did not play a substantial role in the quality of the results.

Table 3 presents the results for mining logs that were generated from the model from Fig. 7a. We generated different logs from this model, in such a way that the duration of activities and the inter-arrival times, were either:

- distributed uniformly with a median that was selected randomly from the interval $(1, 100]$;
- distributed normally with a mean that was selected randomly from the interval $(1, 100]$ and a standard deviation that was selected randomly from the interval of $\frac{1}{5}$ to $\frac{1}{7}$ of the selected mean;
- distributed exponentially with a mean that was selected randomly from the interval $(1, 100]$.

For each of these three distributions, we produced an event log with 200, 2,000, and 10,000 cases, which is the maximum number of cases that can be produced by the BIMP Simulator. In two experiments, the correlation miner was not able to generate an orchestration model, because after removing some variables (e.g. by removing cycles, or P/S threshold filter), it was unable to solve the model with ILP, as the model became infeasible with regard to its constraints. The results in this table show that the correlation miner can mine a simple structured model perfectly, provided that there are sufficiently many cases in the log.



(a) Structured Model with 3 Branches (b) Unstructured Model with 4 Branches

Fig. 7. Evaluation orchestration models

Table 3. Results of using the correlation miner for Fig. 7a

Distribution	Uniform			Exponential			Normal		
# of cases	200	2000	10000	200	2000	10000	200	2000	10000
Recall	1.00	1.00	1.00	n/a	1.00	1.00	n/a	1.00	1.00
Precision	1.00	1.00	1.00	n/a	1.00	1.00	n/a	1.00	1.00

Table 4 presents an overview of the results for mining logs that were generated from the model from Fig. 7b. Experiments on these logs have been conducted with activity duration distributions that were selected in the same manner as for the model from Fig. 7a, but with inter-arrival times that were either selected in the same manner or from an interval with a longer duration ((200, 300]). We also experimented with the ranges (1, 10] and [10, 10]. However, these did not lead to substantially different results. Therefore, due to space restrictions, we do not publish those results here. For each of these combinations of probability distributions, we produced an event log with 100, 1,000, and 10,000 cases.

Under most conditions, the results for this more complex unstructured model are worse than for the structured model. If a structured model contains a task with n outgoing flows, there is for sure another task with n incoming flows; however, if an unstructured model contains a task with n outgoing flows, it may not have any task with n incoming flows. Therefore, solving the structured model is easier for our ILP. The other interesting finding to emerge from Table 4 is that if the inter-arrival time between cases is higher than the service time of activities, this leads to substantially better results. Comparing the first 6 column of results in this table with the second 6, shows that in the experiments with higher inter-arrival time the results for recall and precision noticeably increased on average by 27% and 42% respectively. This result can be explained, because if the inter-arrival time is the same as the service time, cases are more ‘intertwined’ in the log, such that the correlation miner has more difficulties telling them apart based on their timing properties (which other mining algorithms can do based on case identifiers).

Also, as expected, and in line with the findings from Table 3, a higher number of cases in the log leads to better models in most cases. However, in experiments in which activities were distributed exponentially, our algorithm did not find a significant difference between the event logs that originally contained 100 and 1000 cases.

Table 4. Results of using the correlation miner for Fig. 7b

Inter-arrival time	[1–100]						[200–300]					
Distribution	Uniform			Exponential			Uniform			Exponential		
# of cases	100	1000	10000	100	1000	10000	100	1000	10000	100	1000	10000
Recall	0.57	0.79	0.64	0.57	0.57	0.71	0.75	0.93	1.00	0.93	0.71	1.00
Precision	0.42	0.65	0.47	0.42	0.42	0.56	0.86	0.87	1.00	0.81	0.59	1.00

Hence, based on these results, we can conclude that the correlation miner is applicable to mine logs without case identifiers, especially for structured models and when the inter-arrival time between cases is higher than the service time of activities and the number of cases in the log is high enough.

4.3 Real-World Event Log

We also evaluated the algorithm based on a real-world event log. To this end, we used a modified version of the log from the BPI Challenge of 2012 [5]. First, we mined an orchestration model from this event log using Disco [11]. We then removed all loops in order to make it acyclic, because that our algorithm only deals with acyclic models at this stage. For the same reason, we then removed the cases from the log in which loops appeared. The resulting version of this real-world event log contained 13 activities, 70,425 events, and 11,647 cases.

In accordance with the evaluation procedure described in Sect. 4.1, we then removed all case identifiers from the log, mined the log using the correlation miner and evaluated the quality of the mined model. The results obtained from this experiment show a precision of 85% and a recall of 63%. The lower result for recall can be explained by the fact that our algorithm seeks to find edges with higher frequencies, while we used the Disco model that included all possible edges that could be mined from the log, not just the ones with the higher frequency. Normally speaking, one would not be interested in the Disco model that contains all edges, because this model also includes edges that represent exceptional situations. Indeed if we remove edges with lower frequency (i.e., <50) from the original model our recall indeed increases to 68% as precision decreases to 79%.

Based on these results, we claim that the correlation miner is also applicable to mine real-world event logs

5 Related Work

To the best of our knowledge, the closest work to the correlation miner has been published in [8], where the authors tackled the same problem, but based their analysis on event logs neither with case id nor with timestamps. Since the correlation miner also uses timestamps, it is expected to produce more reliable results. Specifically, for the real-world event log (Sect. 4.3), [8] yields a model

with precision 61 % and recall 41 %, which are 24 % and 22 % lower than the correlation miner's results respectively. However, [8] can process cyclic orchestration models.

As suggested in [1], correlating events in an event log is a continuing challenge in the field of process mining, and more specifically, in service mining. Therefore, we discuss the rest of related works in three categories: (i) process discovery, (ii) service mining, and (iii) event correlation.

A large and growing body of literature has been published in the field of process discovery. Van Dongen et al. reviewed discovery algorithms that generate orchestration Petri Net notation [21]. Also, other surveys such as [17], shown that a significant number of studies employ different approaches to discover orchestration models from event logs, such as fuzzy algorithm [10]. A recent study by Verbeek and van der Aalst [22] introduced a generic divide-and-conquer framework to enable process discovery and conformance checking for big event logs. Their approach is similar to our correlation miner in the sense that both have been implemented using Integer Linear Programming. However, the precise optimization problem that they solve is different.

The topic of service mining has been investigated in [2], which illustrated the potential of applying process mining in the context of web services by employing the ProM as a process mining tool and IBM's WebSphere as a reference system. Furthermore, Dustdar et al. [7] introduced Web Services Interaction Mining, which concerns itself with performing process mining techniques in order to analyze service interactions. Finally, [23] presented a web service mining framework aiming at the discovery of unexpected and interesting service compositions. To the best of our knowledge, other than [8], no process or service mining algorithms currently exist that can mine a log that contains no case identifiers and no additional data elements to do the correlation on.

Several techniques have been developed to facilitate event correlation in the context of service-oriented systems. [3] identified a set of 18 correlation patterns that have been grounded in a formal model. The concept of correlation set, a query to retrieve identifiers from messages that are unique for a particular cases, has been included in many correlation techniques such as [9], which proposed an algorithm that assigns a certain identifier to each case in a multi-party supply chains. De Pauw et al. carried out a study to discover conversations in web services by using semantic correlation analysis [4], in which different services pass dedicated identifiers inside their messages. Moreover, in [14] the authors developed an interactive semi-automated tool for event correlation from web service interaction logs. In contrasts, our algorithm enables fully automated event correlation and is only based on occurrences and timestamps of events rather than other data elements that are associated with events.

6 Conclusion

In this paper, we have presented an algorithm, the correlation miner, that enables mining of orchestration models from event logs without case identifiers.

The basic idea of this algorithm is to construct an orchestration model, in such a way that the number of cases that flow into and out of an activity should be equal to the number of events that happen for that activity. However, it is often possible to generate more than one orchestration model that meets this rule. Therefore, we defined additional criteria that need to be fulfilled in order to select the best model. Since logs without case identifiers provide two elements of information: (i) how many times an event occurred for a particular activity, and (ii) at which time an event has occurred, we designed our algorithm based on these two characteristics.

Accordingly, the correlation miner has three steps. In the first two steps, it creates two matrices, the Precede/Succeed matrix and the Duration matrix. Given two activities, the corresponding element in the Precede/Succeed matrix indicates the fraction of events referring to the first activity that have occurred before events referring to the second activity. If this value is high, it is more likely that there is an edge from the first to the second activity. Similarly, given two activities, the corresponding element in the Duration matrix indicates the average time difference between events referring to the first activity and events referring to the second activity. If this value is low, it is more likely that there is an edge from the first to the second activity. Finally, in the third step, the correlation miner constructs all possible orchestration models based that meet the rule mentioned above and then it selects the best one based on the values from the Precede/Succeed matrix and the Duration matrix.

To evaluate the applicability of the correlation miner, we performed experiments with both synthetic event logs and a real-world event log. The results from the evaluation show that the correlation miner produces good results under most conditions. In particular, it produced a model with 85 % precision and 63 % recall for the real-world log. The evaluations with the synthetic logs show that results are better for structured models than for unstructured models. Also, results are better when there are more cases in the log to mine from, and when the inter-arrival time of cases is higher than the duration of activities.

The current correlation miner is only able to mine acyclic orchestrations. There is, therefore, a definite further need for research to enable mining of cyclic orchestrations as well. Furthermore, additional work is needed to ensure that our algorithm can produce an accurate average time difference between activities for the Duration matrix, given that we do not know which events belong to the same case and should, therefore, be used to compute the time difference.

References

1. van der Aalst, W.M.P.: Challenges in service mining: record, check, discover. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 1–4. Springer, Heidelberg (2013)
2. van der Aalst, W.M., Verbeek, H.E.: Process mining in web services: the websphere case. *IEEE Data Eng. Bull.* **31**(3), 45–48 (2008)
3. Barros, A., Decker, G., Dumas, M., Weber, F.: Correlation patterns in service-oriented architectures. In: Dwyer, M.B., Lopes, A. (eds.) FASE 2007. LNCS, vol. 4422, pp. 245–259. Springer, Heidelberg (2007)

4. De Pauw, W., Hoch, R., Huang, Y.: Discovering conversations in web services using semantic correlation analysis. In: ICWS, pp. 639–646. IEEE (2007)
5. van Dongen, B.: Bpi challenge 2012. dataset (2012)
6. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM framework: a new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005)
7. Dustdar, S., Gombotz, R.: Discovering web service workflows using web services interaction mining. *IJBPM* **1**(4), 256–266 (2006)
8. Ferreira, D.R., Gillblad, D.: Discovering process models from unlabelled event logs. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 143–158. Springer, Heidelberg (2009)
9. Gerke, K., Mendling, J., Tarmyshov, K.: Case construction for mining supply chain processes. In: Abramowicz, W. (ed.) Business Information Systems. LNBIP, vol. 21, pp. 181–192. Springer, Heidelberg (2009)
10. Günther, C.W., van der Aalst, W.M.P.: Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 328–343. Springer, Heidelberg (2007)
11. Günther, C., Rozinat, A.: Disco: discover your processes. In: Proceedings of the BPM 2012 Demo Track. CEUR Workshop Proceedings, vol. 940, pp. 40–44 (2012)
12. Johnson, D.B.: Finding all the elementary circuits of a directed graph. *SIAM J. Comput.* **4**(1), 77–84 (1975)
13. Mendling, J., Reijers, H.A., Cardoso, J.: What makes process models understandable? In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 48–63. Springer, Heidelberg (2007)
14. Motahari-Nezhad, H.R., Saint-Paul, R., Casati, F., Benatallah, B.: Event correlation for process discovery from web service interaction logs. *VLDB J.* **20**(3), 417–444 (2011)
15. Optimization, G.: Inc. gurobi optimizer reference manual, version 5.0 (2012)
16. Rozinat, A., van der Aalst, W.M.: Conformance checking of processes based on monitoring real behavior. *Inf. Syst.* **33**(1), 64–95 (2008)
17. Tiwari, A., Turner, C.J., Majeed, B.: A review of business process mining: state-of-the-art and future trends. *BPM J.* **1**, 5–22 (2008)
18. van der Aalst, W.: Process Mining. Springer, Heidelberg (2011)
19. van der Aalst, W.: Service mining: using process mining to discover, check, and improve service behavior. *IEEE Trans. Serv. Comput.* **6**(4), 525–535 (2013)
20. van der Aalst, W., et al.: Process mining manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 169–194. Springer, Heidelberg (2012)
21. van Dongen, B.F., Alves de Medeiros, A.K., Wen, L.: Process mining: overview and outlook of Petri net discovery algorithms. In: Jensen, K., van der Aalst, W.M.P. (eds.) ToPNoC II. LNCS, vol. 5460, pp. 225–242. Springer, Heidelberg (2009)
22. Verbeek, H.M.W., van der Aalst, W.M.P.: Decomposed process mining: the ilp case. In: Fournier, F., Mendling, J. (eds.) BPM 2014 Workshops. LNBIP, vol. 202, pp. 264–276. Springer, Heidelberg (2015)
23. Zheng, G., Bouguettaya, A.: Service mining on the web. *IEEE Trans. Serv. Comput.* **2**(1), 65–78 (2009)

Verification of GSM-Based Artifact-Centric Systems by Predicate Abstraction

Pavel Gonzalez¹, Andreas Griesmayer², and Alessio Lomuscio¹(✉)

¹ Department of Computing, Imperial College London, London, UK
{pavel.gonzalez09,a.lomuscio}@imperial.ac.uk

² ARM, Cambridge, England
andreas.griesmayer@arm.com

Abstract. Artifact-centric systems are a recent paradigm to model and implement business workflows. They describe data, processes, internal and external agents and include mechanisms for data hiding and access control. GSM is a language for the implementation of artifact-centric systems. Since GSM programs have infinitely many states, their verification is challenging. We here present a predicate abstraction technique that enables us to verify GSM programs against rich specifications built on an epistemic, first-order variant of the μ -calculus. We give the theoretical underpinnings of the technique and present GSMC, the first model checker for GSM that implements SMT-based, three-valued abstraction for GSM.

1 Introduction

Artifacts are structures that “combine data and process in an holistic manner” to describe business interactions, typically in a service-oriented architecture [1]. The data component is given by the relational databases underpinning the artifacts in a system, whereas the workflows are described by “lifecycles” associated with each artifact schema. Artifact systems define complex workflow schemes based on artifacts. The system’s participants, or agents, interact with the artifact system by performing events on it.

Differently from services where typically only the process interfaces are advertised, in artifact-centric systems the data structures are also made public. Due to their expressiveness and flexibility, Artifact-centric architectures are increasingly being used in variety of application areas including case management systems [2]. Artifact centric systems are executed in a hub which provides the functionality for service execution. A flexible and powerful language for modelling and executing artifact-centric systems is the Guard-Stage-Milestone programming language (GSM). The open-source design and runtime engine *Acsi Hub* [3,4] is an environment whereby system orchestration and choreography are executed.

If artifact-centric environments are to fulfil their promise to drive the future generation of data-intensive services, they need to be verifiable. This should involve not only the hub itself governing the interactions between artifact calls, but also, and crucially, the agents implementing the services in the system, as is normally done when reasoning about services [5]. In addition to providing

correctness guarantees and rapid prototyping, techniques such as model checking can form the underpinnings for the implementation of automatic service orchestration and choreography [6].

In this paper we develop verification methodologies for artifact-centric systems implemented in GSM. Since GSM programs include data models, they are infinite state programs; it follows that traditional model checking methods based on finite-state machines cannot be applied to them. To address this problem we develop a novel predicate abstraction methodology [7] for GSM defined on a three-valued semantics to account for over- and under-approximation of the models. We also present GSMC, the first model checker for GSM, that implements the technique discussed. We evaluate the technique on a large industrial scale example.

Related Work. Several techniques for the verification of artifact-centric systems have been put forward [8–13]. While these provide considerable insight in the decidability and complexity of the verification problem, they do not provide a concrete verification technique for actual systems. The first contributions concerning the practical verification of GSM systems appeared in [14, 15]. These, however, are defined on coarse, user-given abstractions of GSM models where little data is present and ad-hoc restrictions on variable ranges are applied to obtain finite state systems. Additionally the specification language used is limited.

Incomplete verification methodologies operating directly on the source code have been developed in software verification. The abstraction techniques developed in this context normally target reachability properties only. However, 3-valued abstraction can be applied to specifications based on the μ -calculus [16].

This paper extends existing work by providing 3-valued abstractions for GSM programs specified by a first-order version of the epistemic μ calculus. This enables us to specify services not in purely propositional terms as it is traditionally done but, instead, by referring to the underlying databases.

2 The Guard-Stage-Milestone Language and Multi-agent Systems

While GSM provides a language for the realisation of artifact-centric systems, GSM on its own is not equipped with constructs for the implementation of external actors operating on the system. In GSM these are abstracted by events reaching the system.

However, to verify the possible executions of the system we need to represent how the agents interact with it. Artifact-centric Multi-Agent Systems (AC-MAS) were put forward in [15] to provide a semantics for GSM and the behaviours of external agents. We summarise these concepts below but refer to the cited literature for more details.

The Guard-Stage-Milestone (GSM) has recently been put forward as a declarative language for implementing artifact systems [17]. GSM describes an artifact system Γ that depends on of *artifact types* that correspond to classes of

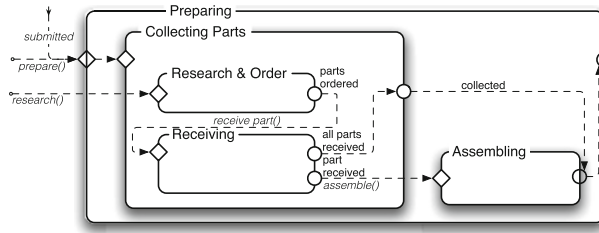


Fig. 1. A lifecycle model.

key business entities. A system comprises of a number of *artifact instances* of artifact types. Each type has an *information model*, which gives an integrated view of the business data, and a hierarchical *lifecycle model*, which describes the structure and evolution of the business process. The artifact system interacts with its environment via *events*. The *information model* is partitioned into the set of *data attributes*, which hold business data, and the set of *status attributes*, which capture the state of the lifecycle model. Figure 1 illustrates a portion of the lifecycle of a manufacturing process and represents the core concepts: The boxes denote *stages*, which represent clusters of activity designed to achieve milestones (o) that represent operational objectives. A *guard* (◊) triggers activities in a stage when a certain condition is fulfilled. Both milestones and guards are controlled declaratively through *sentries*. A sentry of an artifact instance ι is an expression $\chi(\iota)$ in terms of incoming events, guards and milestones, and the status of the instance. In the example above, the Stage ‘Collecting Parts’ contains ‘Research & Order’, which is triggered by an external event; upon reaching the milestone ‘parts ordered’ the next stage ‘Receiving’ is activated.

The operational semantics for GSM is based on the notion of a *business step* (B-step). This is an atomic unit that corresponds to the effect of processing one incoming event. A B-step has the form of a tuple $\sigma = (\Sigma, e, \Sigma')$, where e is an incoming external event and Σ, Σ' are snapshots that capture the current and next state of the information model respectively.

The programming language GSM [4] provides the construct for the realisation of GSM systems; the semantics of GSM programs is given in terms of B-steps.

Artifact-Centric Multi-agent Systems. While GSM models the business artifacts, agents model the possible interactions that external actors and services may have with the artifact system. Below we summarise the key elements from [15] where a formalism for defining the behaviour of the agents, and their access to the artifact system, is described. The concepts of *views* and *windows* are used define which attributes and artifact instances are visible to an agent; *events* represent external actions that cause a change in the system. In the example above, *views* can be used to hide details like procurement of parts from a customer, while allowing access to higher level information, e.g., the start and end of the parts assembling process. Window, instead, can be used to hide orders that do not belong to a particular customer. While a view ν and an event ϵ are

simple lists, a window $\omega_i(\iota)$ is a formula that is evaluated for a specific artifact instance ι and an agent i . The instance is exposed to the agent only if $\omega_i(\iota)$ evaluates to *true*. The behaviour of an agent is given by its *protocol* φ in terms of the visible state of the artifact system, and the agent's unique ID and set of private variables *var*.

We formalise an *agent-based GSM system* for a set of agents \mathcal{A} operating on an environment given by the artifact system E through an Artifact-centric Multi-Agent Systems (AC-MAS) [9]. An AC-MAS $\mathcal{P} = \langle S, \mathcal{I}, Act, \tau, \Lambda \rangle$, where $S \subseteq L_E \times L_1 \times \dots \times L_n$ is the set of *reachable global states*, \mathcal{I} is the *initial state*, $Act = Act_E \times Act_1 \times \dots \times Act_n$ is the set of actions, $\tau : S \times Act \rightarrow 2^S$ is the *global transition relation*, and $\Lambda : S \rightarrow 2^{AP}$ is the *evaluation relation* for a set of propositions AP . A global state $(l_E, l_1, \dots, l_n) \in S$ for the system is given in terms of the snapshot Σ of the artifact system for l_E , and the accessible variables of each agent for l_1, \dots, l_n . We also write $l_i(s)$ to extract the visible state for agent i from a global state $s \in S$. The sets of actions Act_E and Act_i are directly defined by the events the system provides and the permissions of the agents. The global transition relation $\tau(s, \alpha)$ with $s \in S$ and $\alpha \in Act$ is given by the corresponding B-steps defined by GSM in combination with the protocols φ of the agents, where only one agent can interact with the artifact system at a time while the others are idle.

The initial state \mathcal{I} is a global state with not artifact instances in Σ and with all private variables set to their initial value. We write $s \rightarrow s'$ iff there exists an α , such that $s' \in \tau(s, \alpha)$; in this case s' a *successor* of s . A *run* r from s is an infinite sequence $s^0 \rightarrow s^1 \rightarrow \dots$ with $s^0 = s$. We write $r[i]$ for the i -th state in the run and r_s for the set of all runs starting from s . A state s' is *reachable* from s if there is a run from s that contains s' , formally $\exists r' \in r_s : \exists i \geq 0 : r'[i] = s'$. Note that portions of the global state may not be visible to an agent. In line with the standard semantics of epistemic logic [18], we say that the states s and s' are *epistemically indistinguishable* for agent i , or $s \sim_i s'$, iff $l_i(s) = l_i(s')$, i.e., if agent i 's local state is the same in s and s' .

3 Three-Valued Abstraction for AC-MAS

Predicate Abstraction [7] is a technique used to generate sound approximations of infinite state systems by grouping together system states satisfying certain properties into *abstract states*. *May* transition between abstract states correspond to possible transitions between some of corresponding concrete states. This leads to an over-approximation of the possible behaviour that is *conservative* for safety properties but may lead to unsound results otherwise. Three-valued abstraction has been employed [16, 19] to overcome these limitations. In three-valued abstraction a second transition relation (or *must* relation) is introduced to encode when a change in the corresponding concrete states must happen. This allows to concurrently maintain over- and under-approximations that are conservative for both positive and negative specifications and allows to detect when a result cannot be determined.

To extend this technique to AC-MAS, we introduce the three-valued semantics for the epistemic μ -calculus and replace τ with τ_m , the *global may transition relation*, and τ_M , the *global must transition relation*, to get $\mathcal{P} = \langle S, \mathcal{I}, Act, \tau_m, \tau_M, \Lambda \rangle$. Analogously to the concrete case, we write $s \xrightarrow{\text{may}} t$ ($s \xrightarrow{\text{must}} t$) for $t \in \tau_m(s, a)$ ($t \in \tau_M(s, a)$). Over- and under-approximations for the epistemic relations are denoted as \sim_i^{may} and \sim_i^{must} respectively. This extended definition of AC-MAS allows us to define abstraction formally as:

Definition 1 (Abstraction). *Let $\mathcal{P} = \langle S, \mathcal{I}, Act, \tau_m, \tau_M, \Lambda \rangle$ and $\mathcal{P}' = \langle S', \mathcal{I}', Act', \tau'_m, \tau'_M, \Lambda' \rangle$ be AC-MAS over the same set \mathcal{A} of agents and sets $AP' \subseteq AP$ of propositions. We say that \mathcal{P}' is an abstraction of \mathcal{P} if:*

1. $s' \in \mathcal{I}'$ iff there exists $s \in \mathcal{I}$, such that $s \in \gamma(s')$;
2. $s' \xrightarrow{\text{may}'} t'$ iff there exist $s \in \gamma(s')$ and $t \in \gamma(t')$, such that $s \xrightarrow{\text{may}} t$;
3. $s' \xrightarrow{\text{must}'} t'$ iff for each $s \in \gamma(s')$ there exists $t \in \gamma(t')$, such that $s \xrightarrow{\text{must}} t$;
4. $s' \sim_i^{\text{may}'} t'$ iff there exist $s \in \gamma(s'), t \in \gamma(t')$ such that $s \sim_i^{\text{may}} t$ or there exists u' such that $s' \sim_i^{\text{may}} u'$ and $u' \sim_i^{\text{may}} t'$;
5. $s' \sim_i^{\text{must}'} t'$ iff for each $s \in \gamma(s')$ there exists $t \in \gamma(t')$, such that $s \sim_i^{\text{must}} t$, and for each $t \in \gamma(t')$ there exists $s \in \gamma(s')$, such that $t \sim_i^{\text{must}} s$;
6. $p \in \Lambda'(s')$ iff $p \in \Lambda(s)$ for each $s \in \gamma(s')$;

where $\gamma : S' \mapsto 2^S$ is the concretisation function that maps each abstract state $s' \in S'$ to the non-empty set of concrete states $S_{s'} \subseteq S$ it represents; $\xrightarrow{\text{may}'}'$ and $\xrightarrow{\text{may}}$ are the may transition relations in \mathcal{P}' and \mathcal{P} respectively; $\xrightarrow{\text{must}'}'$ and $\xrightarrow{\text{must}}$ are the must transition relations; $\sim_i^{\text{may}'}'$ and \sim_i^{may} are the may epistemic relations; and $\sim_i^{\text{must}'}'$ and \sim_i^{must} are the must epistemic relations.

May transition relations in the abstract model \mathcal{P}' over-approximate may transition relations in the concrete model \mathcal{P} : whenever there is a may transition between two states in \mathcal{P} , there is a transition between the corresponding abstract states of \mathcal{P}' . Conversely, must transition relations in the abstract model \mathcal{P}' under-approximate must transition relations in the concrete model \mathcal{P} ; they are only created for concrete transitions that are common to all of the states of \mathcal{P} represented by the source abstract state.

We define may and must epistemic possibility relations in the abstract system similarly to the temporal case; however, there are additional constraints due to the nature of the relations. Specifically, we require both to be equivalence relations. This is achieved by building the transitive closure for $\sim_i^{\text{may}'}'$, while relations in $\sim_i^{\text{must}'}'$ that are not symmetric are removed. By insisting on equivalence relations, we ensure that the usual KT45 axioms [18] for knowledge are satisfied in the abstract model.

Note that if the abstract may epistemic possibility relation were defined analogously to abstract may transition relations, it would not necessarily be transitive. Therefore, we define the abstract may epistemic possibility relation as the transitive closure of this relation. Similarly, if the abstract must epistemic possibility relation were defined analogously to abstract must transition relations, it would not be necessarily symmetric. Therefore, we remove the abstract

must epistemic possibility relations that are not symmetric. The labelling of an abstract state is defined so that it is consistent with the labelling of all the concrete states it represents. The bi-implication ensures that the abstract labelling function is *exact*.

We use an extension of the epistemic μ -calculus [20] as our specification language. We use the observational semantics for the epistemic component K_i in addition to the standard μ -calculus [21] and define the language \mathcal{L} in BNF notation as follows. Let AP be a finite set of atomic propositions and \mathcal{V} a set of propositional variables, then:

$$\varphi ::= \top \mid p \mid Z \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Box\varphi \mid K_i\varphi \mid \mu Z.\varphi \mid \nu Z.\varphi$$

where $p \in AP$ and $Z \in \mathcal{V}$. Here $K_i\varphi$ means *agent i knows φ* [18].

The syntactic combinations μZ and νZ are the *least* and *greatest fix-point operators* respectively. An *interpretation* $\rho : \mathcal{V} \rightarrow 2^S$ assigns the free propositional variable Z as a set of states. Any occurrence of Z in φ falls within an even number of negations. Furthermore, we assume that formulas are *closed* and *well-named*, i.e., all propositional variables are bound exactly once in any formula.

To evaluate a formula φ , we compute sets of states such that a state s satisfies φ if $s \in \llbracket \varphi \rrbracket_{tt}^{\mathcal{P},\rho}$; a state s refutes φ if $s \in \llbracket \varphi \rrbracket_{ff}^{\mathcal{P},\rho}$. In addition to satisfaction (tt) and refutation (ff), we write \perp to express that the truth value is unknown. We define the three-valued semantics for \mathcal{L} in line with [16] and extend it by the epistemic operator K_i as follows:

Definition 2 (Three-Valued Semantics). *Let \mathcal{P} be AC-MAS. The three-valued semantics of $\varphi \in \mathcal{L}$ in \mathcal{P} for an environment ρ , denoted $\llbracket \varphi \rrbracket_3^{M,\rho}$, is defined by a mapping $S \rightarrow \{tt, ff, \perp\}$ such that:*

$$\llbracket \varphi \rrbracket_3^{\mathcal{P},\rho}(s) = \begin{cases} tt, & \text{if } s \in \llbracket \varphi \rrbracket_{tt}^{\mathcal{P},\rho} \\ ff, & \text{if } s \in \llbracket \varphi \rrbracket_{ff}^{\mathcal{P},\rho} \\ \perp, & \text{otherwise} \end{cases}$$

The sets $\llbracket \varphi \rrbracket_{tt}^{\mathcal{P},\rho} \subseteq S$ and $\llbracket \varphi \rrbracket_{ff}^{\mathcal{P},\rho} \subseteq S$ for $\varphi \in \mathcal{L}$ over \mathcal{P} are defined as:

$$\begin{array}{ll} \llbracket \top \rrbracket_{tt}^{\mathcal{P},\rho} = S & \llbracket \top \rrbracket_{ff}^{\mathcal{P},\rho} = \emptyset \\ \llbracket p \rrbracket_{tt}^{\mathcal{P},\rho} = \{s \in S : p \in \Lambda(s)\} & \llbracket p \rrbracket_{ff}^{\mathcal{P},\rho} = \{s \in S : p \notin \Lambda(s)\} \\ \llbracket Z \rrbracket_{tt}^{\mathcal{P},\rho} = \rho(Z) & \llbracket Z \rrbracket_{ff}^{\mathcal{P},\rho} = \rho(Z) \\ \llbracket \neg\varphi \rrbracket_{tt}^{\mathcal{P},\rho} = \llbracket \varphi \rrbracket_{ff}^{\mathcal{P},\rho} & \llbracket \neg\varphi \rrbracket_{ff}^{\mathcal{P},\rho} = \llbracket \varphi \rrbracket_{tt}^{\mathcal{P},\rho} \\ \llbracket \varphi_1 \wedge \varphi_2 \rrbracket_{tt}^{\mathcal{P},\rho} = \llbracket \varphi_1 \rrbracket_{tt}^{\mathcal{P},\rho} \cap \llbracket \varphi_2 \rrbracket_{tt}^{\mathcal{P},\rho} & \llbracket \varphi_1 \wedge \varphi_2 \rrbracket_{ff}^{\mathcal{P},\rho} = \llbracket \varphi_1 \rrbracket_{ff}^{\mathcal{P},\rho} \cup \llbracket \varphi_2 \rrbracket_{ff}^{\mathcal{P},\rho} \\ \llbracket \Box\varphi \rrbracket_{tt}^{\mathcal{P},\rho} = ax(\llbracket \varphi \rrbracket_{tt}^{\mathcal{P},\rho}) & \llbracket \Box\varphi \rrbracket_{ff}^{\mathcal{P},\rho} = ex(\llbracket \varphi \rrbracket_{ff}^{\mathcal{P},\rho}) \\ \llbracket \mu Z.\varphi \rrbracket_{tt}^{\mathcal{P},\rho} = \text{lfp}(\lambda g.\llbracket \varphi \rrbracket_{tt}^{\mathcal{P},\rho[Z \mapsto g]}) & \llbracket \mu Z.\varphi \rrbracket_{ff}^{\mathcal{P},\rho} = \text{gfp}(\lambda g.\llbracket \varphi \rrbracket_{ff}^{\mathcal{P},\rho[Z \mapsto g]}) \\ \llbracket \nu Z.\varphi \rrbracket_{tt}^{\mathcal{P},\rho} = \text{gfp}(\lambda g.\llbracket \varphi \rrbracket_{tt}^{\mathcal{P},\rho[Z \mapsto g]}) & \llbracket \nu Z.\varphi \rrbracket_{ff}^{\mathcal{P},\rho} = \text{lfp}(\lambda g.\llbracket \varphi \rrbracket_{ff}^{\mathcal{P},\rho[Z \mapsto g]}) \\ \llbracket K_i\varphi \rrbracket_{tt}^{\mathcal{P},\rho} = ax_i(\llbracket \varphi \rrbracket_{tt}^{\mathcal{P},\rho}) & \llbracket K_i\varphi \rrbracket_{ff}^{\mathcal{P},\rho} = ex_i(\llbracket \varphi \rrbracket_{ff}^{\mathcal{P},\rho}) \cup \llbracket \varphi \rrbracket_{ff}^{\mathcal{P},\rho} \end{array}$$

where for $X \subseteq S$: $ax(X) = \{s \mid \forall s' : s \xrightarrow{\text{may}} s' \Rightarrow X\}$, $ex(X) = \{s \mid \exists s' : s \xrightarrow{\text{must}} s' \wedge X\}$, $ax_i(X) = \{s \mid \forall s' : s \xrightarrow{\text{may}}_i s' \Rightarrow X\}$, and $ex_i(X) = \{s \mid \exists s' : s \xrightarrow{\text{must}}_i s' \wedge X\}$. Intuitively, ax returns states whose may successors are all in X . In contrast, ex computes all states for which at least one must transition exists. Similarly, ax_i and ex_i are the corresponding operators for the epistemic relations for a given agent i and give the set of the respective indistinguishable states. The definition for $\llbracket K_i \varphi \rrbracket_{\text{ff}}^{\mathcal{P}, \rho}$ allows for a tighter under-approximation since agents do not know φ in states where φ is false.

An AC-MAS \mathcal{P} satisfies a formula φ , or $[\mathcal{P} \stackrel{3}{=} \varphi] = \text{tt}$, if all its initial states are in $\llbracket \varphi \rrbracket_{\text{tt}}^{\mathcal{P}, \rho}$. An AC-MAS \mathcal{P} refutes φ , or $[\mathcal{P} \stackrel{3}{=} \varphi] = \text{ff}$, if at least one initial state is in $\llbracket \varphi \rrbracket_{\text{ff}}^{\mathcal{P}, \rho}$. Otherwise we say $[\mathcal{P} \stackrel{3}{=} \varphi] = \perp$. Note that the abstraction for AC-MAS models \mathcal{P} as defined above is *consistent*, i.e., $\llbracket \varphi \rrbracket_{\text{tt}} \cap \llbracket \varphi \rrbracket_{\text{ff}} = \emptyset$ for any $\varphi \in \mathcal{L}$. Therefore the set $\llbracket \varphi \rrbracket_{\perp}^{\mathcal{P}, \rho}$ can be computed as $S \setminus (\llbracket \varphi \rrbracket_{\text{tt}}^{\mathcal{P}, \rho} \cup \llbracket \varphi \rrbracket_{\text{ff}}^{\mathcal{P}, \rho})$.

Abstracting GSM. To instantiate the theory above, we now outline a methodology for constructing *abstract* AC-MAS models from *concrete* GSM programs. This process includes abstracting the data to build a finite model using *predicates*, as well as the computation of the temporal and epistemic *may* and *must* relations. Observe that GSM programs only regulate the evolution of the artifact-centric system in the presence of external events and do not include a description of the agents' behaviour with the system. To account for the evolution of both we combine GSM programs with procedural agent descriptions, thereby obtaining a GSM-MAS program. We do not present the agents descriptions here; we simply assume that they define the local states for the agents and define their evolution, both in terms of the actions performed on the artifact-centric system (or events) and the changes to their local state in the presence of actions. By GSM-MAS we refer to the combined programs consisting of the GSM code and the agents descriptions. It can be checked that AC-MAS provide a semantics for GSM-MAS programs.

Given a GSM-MAS program \mathcal{P} and a specification φ as input, we generate an abstract \mathcal{P}' such that if checking $\mathcal{P}' \models \varphi$ returns either *true* or *false*, then the same result also applies to \mathcal{P} ; if $\mathcal{P}' \models \varphi$ returns undefined, then no conclusion can be drawn on \mathcal{P} and the abstraction needs to be refined.

States in the abstract system are represented by *predicates*, which are Boolean variables that represent the validity of expressions in the concrete system. Predicates are selected by analysing the GSM-MAS program and the specification to be verified. In doing so we retain the status attributes of the lifecycles, as these are already Boolean, but replace the potentially unbound data attributes. To capture key conditions in the system, binary relations ($=, \neq, <, \leq, >, \geq$) or quantifications over sets of data (\exists, \forall) are selected by syntactically analysing the GSM-MAS program to get an initial set of predicates p_i .

In contrast to classical approaches, which build abstractions locally to single execution blocks, the declarative nature of GSM-MAS programs and the quantification over artifact instances results in predicates that are shared between instances or agents. While predicates that are *local* to an artifact instance or agent can be treated as instance variables, *shared* predicates need to be treated

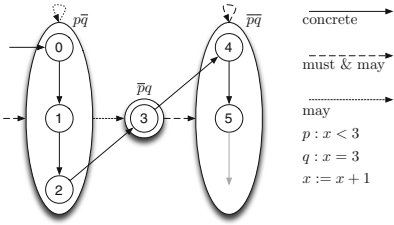


Fig. 2. Concrete and abstract transitions of a non-negative integer counter.

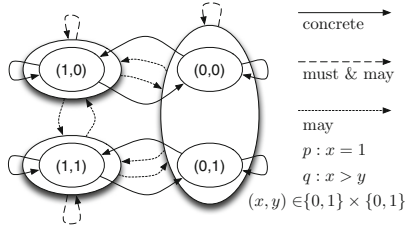


Fig. 3. Indistinguishable states of an agent given $y \in \nu$.

carefully to avoid incorrect abstractions for the local states of the agents. Building the abstract state using data predicates along with the original status attributes guarantees that the abstract system retains the same structure, while maintaining an over-approximation of the data space of the concrete system.

Since several concrete states correspond to an abstract state, temporal changes in the abstract system can only approximate the corresponding changes in the concrete data. Rather than giving the full procedure, instead we here compute the may and must transition relations on a simple example. Consider the abstraction of a non-negative integer counter with a single integer variable x that is initialised to 0 and gets incremented by 1 at each step using the assignment $x := x + 1$. If we base our abstract states on the predicates $p : x < 3$ and $q : x = 3$, we have three possible abstract states, which are shown in Fig. 2. Between the abstract states $p\bar{q}$ and $\bar{p}q$ we have a *may* transition because the concrete system *can* transition to a state that is in $\bar{p}q$. There is no *must* transition, however, because from a state in $p\bar{q}$ the concrete system can also transition to a state that is still in $p\bar{q}$. In contrast, all concrete states in $\bar{p}q$ transition to $\bar{p}q$, which means that we have both may and must transitions.

In line with existing literature in epistemic logic [18], the agents’ knowledge is computed on the basis of the equality of their local components. In our case, however, the agents’ local states are given by private variables, but also their *view* ν and the *window* ω . In the labelling algorithm for computing the sets in which an epistemic formula holds, the existential pre-image $\sim_i(X)$ of the set of global states X with respect to the appropriate epistemic relation (\sim_i^{may} or \sim_i^{must}) is computed by existential quantification of variables outside of the view, and restriction to the window. The pre-image can be directly used to compute $\llbracket K_i\varphi \rrbracket_{\text{ff}}^{\mathcal{P},\rho}$, since $\sim_i^{\text{must}}(\llbracket \varphi \rrbracket_{\text{ff}}^{\mathcal{P},\rho}) = ex_i(\llbracket \varphi \rrbracket_{\text{ff}}^{\mathcal{P},\rho}) = \{s \mid \exists s' : s \sim_i^{\text{must}} s' \wedge \llbracket \varphi \rrbracket_{\text{ff}}^{\mathcal{P},\rho}\}$. This is not the case for $\llbracket K_i\varphi \rrbracket_{\text{tt}}^{\mathcal{P},\rho}$, where $\sim_i^{\text{may}}(X) = \{s \mid \exists s' : s \sim_i^{\text{may}} s' \wedge X\}$; in this case we first compute the pre-image of $\llbracket \varphi \rrbracket_{\text{ff}}^{\mathcal{P},\rho}$ and then take its complement.

To build the abstract epistemic relations, views and windows have to be defined in terms of the predicates for the abstract states. The window ω can be expressed as a formula using relations between variables. Since we build our set of predicates using exactly those relations, we can build a direct mapping to an

abstract function ω' . In other words, the abstract and concrete window functions represent the exact same states and $\omega(\gamma(x)) = \omega'(x)$ for any abstract state x .

The abstraction of the view ν is less straightforward, however, as predicates may use sets of variables that do not coincide with ν , and in the case of shared predicates may even relate to different instances and agents. This implies that an agent may be able to determine the value of a predicate only for some states. To avoid computing ν' depending on the state, we compute two sets ν_{may} and ν_{must} that give correct over- and under-approximations of the epistemic relation.

For the over-approximation $\overset{\text{may}}{\sim}_i$, we select only the *local* predicates for ν_{may} that exclusively refer to visible variables in ν . This ensures that an agent can distinguish two states in the abstract system only if it has enough visibility in the concrete system to determine the value of the predicates. We exclude *shared* predicates since one or more of the referenced instances might be outside the window ω and thus the predicate may be unknown. Note that fewer predicates in ν result in a larger set $\sim_i(X)$, thereby ensuring that an over-approximation is generated. This set is then restricted to the set R_{may} of reachable states computed with $\overset{\text{may}}{\rightarrow}$, which represent the states possibly reachable in the abstract model.

For the must transitions $\overset{\text{must}}{\sim}_i$, we need to ensure under-approximation; we stipulate that $s \overset{\text{must}}{\sim}_i t$ if for each of the concrete states in s there is a concrete state in t such that there is an epistemic relation for agent i between them. Intuitively, this means that we need to consider every predicate for ν_{must} that encodes at least one variable visible in the concrete system. Note, however, that this may not be sufficient as, if the predicates are not independent of each other, they may allow to infer information about a value even if it is not visible to the agent. Consider the example in Fig. 3 with $p : x = 1$ and $q : x > y$ with the visible variable y . In the concrete system, $(x, y) = (1, 1)$ is distinguishable from $(0, 0)$, but not from $(0, 1)$. To compute $\overset{\text{must}}{\sim}_i$ with visible predicate q and only quantify p would result in a transition between $p\bar{q}$ and $\bar{p}q$, which is not a proper under-approximation because of the missing epistemic relation between $(0, 0)$ and $(1, 1)$ in the concrete system. To ensure a correct under-approximation is generated, we transitively select all predicates that share the variables with predicates already in ν_{must} and also include *shared* predicates. Finally, we restrict $\overset{\text{must}}{\sim}_i$ by R_{must} , computed by $\overset{\text{must}}{\rightarrow}$, which corresponds to the set of states that are known to be reachable in the concrete system.

4 Implementation and Experimental Results

GSMC is an open source model checker that implements the technique described above [22]. It is operated via a command line application written in C++ that uses the CUDD library [23] for BDD operations and the SMT solver CVC4 [24] to help compute the abstractions. GSMC uses binary decision diagrams (BDDs) to represent the sets of states and the transition relations of the abstract model.

GSMC operates directly on GSM programs developed in the *Acsi Hub* [4], a web-based application that supports the design and implementation of artifact systems. By using the *Acsi Hub*, users can design business artifacts with GSM

lifecycles through a design editor and then immediately deploy these programs on an execution engine. The description of the agents and specification properties are supplied in plain text files.

GSMC supports specifications written in a temporal-epistemic logic with quantification over artifact instances. The language, called *Instance Quantified CTLK* [15], or IQ-CTLK, extends the usual epistemic branching time logic CTLK and has the following syntax:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\varphi \mid EG\varphi \mid E(\varphi U \varphi) \mid K_i\varphi \mid \forall x : R \varphi \mid \exists x : R \varphi$$

where R is the name of an artifact type and p is an atomic proposition over the agents' private data and the attributes of active instances that are specified in terms of *instance variables* bound by the quantification operators. The quantified instance variables range over the active instances of a given artifact type R in the state where the quantification is evaluated and must be bound.

We introduce a *bound* on the number of instances that can be generated and use an *overflow* flag that indicates if the bound was reached during a run. The bound in the number of instances restricts the possible behaviour of the system and may lead to loss of soundness or completeness when the limit is reached. The bound can be revised before any execution. Any IQ-CTLK formula to be verified is first rewritten into a CTLK formula by replacing the quantification operators as follows:

$$\forall x : \varphi \Rightarrow \bigwedge_{\iota \in \Gamma} \text{created}(\iota) \rightarrow \varphi \qquad \exists x : \varphi \Rightarrow \bigvee_{\iota \in \Gamma} \text{created}(\iota) \wedge \varphi$$

where the expression $\text{created}(\iota)$ checks if instance ι was created. This is required since the new formula ranges over the actual instances, which are created dynamically at run-time, and the number of *active* instances is not a priori known. The CTLK formula is then translated to an epistemic μ -calculus formula using the fixed point characterisation of CTL [25]; the resulting specification is checked on the abstract model.

In the rest of the section we evaluate the tool. Both use cases are complete **Acsi Hub** applications. We verify the temporal-epistemic properties of the systems and discuss performance of the implemented techniques. All tests were conducted on a 64-bit Fedora 17 Linux machine with a 2.10 GHz Intel Core i7 processor and 4 GB RAM.

Evaluation: The Order-to-Cash Scenario. This is an application in which a seller schedules the assembly of a product based on a confirmed purchase order from a buyer that requires several components, that are sourced from different suppliers. When the product is assembled, a carrier ships the order to the buyer. The buyer can cancel a purchase order at any time before the delivery. We refer to [17] for more details. The GSM program consists of a single-artifact **Acsi Hub** application with 10 data attributes, 9 stages, 11 milestones, and 12 events. We model a collection of components by introducing an integer counter. The process is considered complete when 3 components have arrived. The following three

agent roles interact with the artifact system: (1) a *Buyer* who creates an artifact instance that represents the order; (2) a *Seller* who fulfils the order; and (3) a *Carrier* who ships the finished product to the Buyer.

We constructed several GSM-MAS with different numbers of agents and bounds on artifact instances. We report on the verification of these systems against four temporal-epistemic specifications. In the following *Diogenes* is an agent of role *Buyer*. The first specification, Property 1, states that *Diogenes* knows that the product might be received via any of his orders as long as these are not cancelled, i.e., that there is no deadlock in processing the order:

$$AG \forall x : CustomerOrder((x.BuyerId \neq Diogenes \wedge \neg Diogenes.Cancelled) \rightarrow K_{Diogenes} EF x.Received) \quad (1)$$

Property 2 states that *Diogenes* may come to know that a product is received for an order with a different owner. This can be used to ascertain whether the orders are private to the buyers:

$$EF \exists x : CustomerOrder(x.BuyerId \neq Diogenes \wedge K_{Diogenes} x.Received) \quad (2)$$

Property 3 encodes the ability of an agent to deduce information it can not directly observe by checking whether *Diogenes* always knows there are 3 *PurchaseOrders* collected in all of his orders when the milestone *Ready* is achieved:

$$AG \forall x : CustomerOrder((x.Ready \wedge x.BuyerId = Diogenes) \rightarrow K_{Diogenes} (x.PurchaseOrders = 3)) \quad (3)$$

The last specification, Property 4, encodes the ownership of the order. It implies that an agent other than *Diogenes* can cancel an order that belongs to *Diogenes*. This is done by using a private variable, which is true only if *Diogenes* executed the *Cancelled* event. We thus require that an order that belongs to *Diogenes* cannot be cancelled if this variable is false:

$$EF \exists x : CustomerOrder(x.BuyerId = Diogenes \wedge x.Cancelled \wedge Diogenes.cancelled \neq 1) \quad (4)$$

We first verified the properties in the abstract system and measured the number of may and must reachable states, memory used, and CPU time required. GSMC evaluated Property 1 to be *unknown*, Properties 2 and 4 to be *false*, and Property 3 to be *true* in the abstract model. Table 1 reports the performance for a system with 1 agent per role and a system of 15 agents (6 Buyers, 5 Sellers, and 4 Carriers). We observe that there is an order of magnitude of difference in the number of may and must reachable states; this implies that there are specifications, such as Property 1, that cannot be determined. However, the tool is still able to find answers to the other three properties. The results are in line with our expectations, confirming the correctness of the GSM program against said specifications.

Table 1. Performance for different numbers of artifact instances ι and agents.

# ι	3 agents				15 agents			
	#may	#must	MB	s	#may	#must	MB	s
1	0.91 e2	0.45 e2	55	0.2	1.65 e3	5.89 e2	69	2.1
2	2.23 e3	5.27 e2	78	0.9	1.32 e6	1.55 e5	106	4.6
3	5.34 e4	5.45 e3	93	4.8	1.03 e9	3.83 e7	124	31.9
4	1.28 e6	5.46 e4	112	25.5	7.99 e11	9.02 e9	233	168.8
5	3.10 e7	5.42 e5	172	90.4	6.05 e14	2.05 e12	463	596.2
6	7.57 e8	5.36 e6	273	257.2	4.53 e17	4.57 e14	898	2014.2

For a comparison we disabled the predicate abstraction feature and verified the same Order-to-Cash system under the same conditions. In this case GSMC evaluated Properties 1 and 3 to be *true* and Properties 2 and 4 to be *false* in the model, which is consistent with the abstraction results. Note that the previously unknown Property 1 is returned as *true* when predicate abstraction is disabled.

Table 2 presents the performance of the tool executed on the same machine, under the same conditions. By comparing this table to Table 1, we see that verification of the concrete model initially outperforms abstraction. This is because there is a constant overhead from building the may and must temporal transitions by calls to the SMT solver. However, as the model grows we clearly see the benefits of the abstraction methodology as it reduces the number of states to be considered. For example, for 15 agents and 5 instances we have over two orders of magnitude reduction in the number of states to be considered and an order of magnitude reduction in the verification time.

Although the tool does not support automatic refinement for the abstraction methodology, by manually adding the predicates $x.PurchaseOrders = 0$, $x.PurchaseOrders = 1$, and $x.PurchaseOrders = 2$ we could refine the abstract model in such a way that may and must reachable state spaces become equal to those of the concrete model. In doing so Property 1 is no longer returned as unknown but true; this is in line with the results obtained by verifying the concrete system.

The Second Evaluation Scenario focuses on the management of research programs. The scenario consists of three conceptual entities modelled as business artifacts: *CallForProposals* represents the annual call of a funding program; *Project* encodes one project which starts as a proposal and, if successful, becomes a funded research project; *ReviewBoard* governs the assembling of a review board for a specified research topic and the reviews of all competing proposals. We focus on three roles: the *Program Manager* initiates the process and confirms the board; the *Program Staff Member* supervises projects on behalf of the funding agency, the *Project Leader* is responsible for a particular proposal. The scenario was implemented in the *Acsi Hub*. We refer to [26] for detail.

Table 2. Performance for different settings of the concrete system.

# ι	3 agents			15 agents		
	#states	MB	s	#states	MB	s
1	1.17 e2	27	0.1	2.92 e3	31	0.2
2	3.71 e3	52	0.7	4.16 e6	70	4.9
3	1.16 e5	64	5.9	5.82 e9	84	65.5
4	3.67 e6	96	42.1	8.01 e12	222	360.2
5	1.18 e8	195	176.7	1.09 e16	539	1419.6
6	3.83 e9	375	500.5	N/A	N/A	N/A

The GSM program for this scenario is a significantly larger application than the Order-to-Cash, as it consists of 45 stages, 56 milestones, and 19 events. For this reason we here report only the interactions between the agents and the *ReviewBoard* artifact type only, i.e., the types *CallForProposals* and *Project* are not analysed here. We also restrict the number of agents to one per role. Nevertheless, GSMC builds the transition relations for the *whole* GSM program.

An artifact instance is created when the agent *Manager* decides to set up a review board. When the *Manager* confirms the assembled board, the lifecycle of the *ReviewBoard* instance terminates. The agent *Staff* carries out several administration task, including assembling and updating the review board. Both *Manager* and *Staff* can access all artifact instances. In contrast, the agent *Leader* cannot observe any of them. Agents do not set specific payloads; this implies we can examine all the possible non-deterministic behaviours.

The first two specifications we analyse concern the simple reachability of stages and milestones. Property 5 states that there is an instance of the *ReviewBoard* artifact type in which eventually the stage *SendProposalsToReviewers* is open:

$$EF \exists x : ReviewBoard(x.SendProposalsToReviewers) \quad (5)$$

Property 6 encodes that there is an instance of *ReviewBoard* in which eventually the milestone *ReviewsTerminated* is achieved. This means that an instance will terminate:

$$EF \exists x : ReviewBoard(x.ReviewsTerminated) \quad (6)$$

The next two specifications demonstrate the use of 3-valued abstraction on sets of data. These formulas cannot be verified on concrete systems as sets of data cannot be represented on concrete models. Property 7 states that there is an instance of *ReviewBoard* in which eventually the the active reviewers is equal to the specified number of reviewers required:

$$EF \exists x : ReviewBoard(x.Reviewers.size() = x.ReviewBoardSize) \quad (7)$$

Property 8 states that there is an instance of *ReviewBoard* in which eventually the set of active reviewers contains a reviewer called *Diogenes*:

$$EF \exists x : ReviewBoard(x.Reviewers.exists(FirstName = Diogenes)) \quad (8)$$

Table 3. Performance results for 1 instance of the *ReviewBoard* artifact type.

Operation	Result	Memory	Time
Computation of τ_m and τ_M	✓	395 MB	33.16 s
Computation of R_{may} and R_{must}	✓	364 MB	3.06 s
Property 5	✓	280 MB	1.21 s
Property 6	✓	284 MB	1.02 s
Property 7	✓	278 MB	0.80 s
Property 8	✓	272 MB	0.92 s
Property 9	✓	312 MB	1.54 s
Property 10	✗	320 MB	2.22 s

The last two specifications concern reasoning about the knowledge of the agents. Property 9 says that agent *Manager* knows there is a path where eventually the milestone *ReviewsTerminated* is achieved:

$$K_{Manager} (EF \exists x : ReviewBoard(x.ReviewsTerminated)) \quad (9)$$

Finally, Property 10 encodes that agent *Leader* knows there is a path where eventually the milestone *ReviewsTerminated* is achieved:

$$K_{Leader} (EF \exists x : ReviewBoard(x.ReviewsTerminated)) \quad (10)$$

The data attributes of the concrete model are represented by 10 predicates in the abstract model. The abstract model is then encoded by GSMC into BDDs by using 142 Boolean variables. As the construction of the transition relations requires three distinct sets of Boolean variables, there are 426 Boolean variables in total. The *may* reachable state space of the model spans over approximately 7.1×10^9 states, and its construction requires 30 iterations. The *must* reachable state space has 8.4×10^7 states and it is built in 12 iterations. The total time for the verification was 43.88s and the memory usage peaked at 395 MB.

Table 3 presents the performance of the individual operations undertaken by GSMC, as well as the verification results. The first row reports the construction of the transition relations, the second row shows the construction of may and must reachable state spaces, and the remaining rows give the performance for the properties verified in this section. Properties 5–9 are *true* in the model. Property 10 is *false* in the model since the agent *Leader* cannot observe the *ReviewBoard* lifecycle.

5 Conclusions

Artifact-centric systems have been put forward as an intuitive paradigm to model applications for businesses and services. Differently from process models, artifact-centric systems give equal prominence to both the process model

(i.e., the lifecycles) and that information model (i.e., the data structures). GSM has been introduced as a programming framework for artifact-centric systems and recently adopted as part of the OMG Case Management Model and Notation standard [27]. This suggests its use may increase considerably in the future.

In this paper we introduced a methodology for the verification of GSM systems. The technique extends state-of-the-art methods in verification by providing a predicate abstraction methodology to GSM. In addition to catering for GSM programs directly, we support first-order quantification to refer to the data referenced by artifacts. Differently from any other mainstream predicate abstraction technique we also support operators expressing the knowledge of the agents in the system.

We implemented the technique in GSMC, the first model checker for GSM that supports GSM's information model. The checker supports GSM's infinite models and automatically generates, via SMT calls, finite abstract models that can be efficiently encoded as BDDs and then verified. To evaluate the efficiency of the approach we have discussed the experimental results obtained by using the checkers on sophisticated use-cases generated by third-parties in the EU project ACSI. The approach as currently implemented does not support recursion in the GSM programs. In the future we plan to add partial support for basic recursive data types and automatic refinement.

References

1. Cohn, D., Hull, R.: Business artifacts: a data-centric approach to modeling business operations and processes. *Bull. IEEE Comput. Soc. Tech. Committee Data Eng.* **32**(3), 3–9 (2009)
2. Marin, M., Hull, R., Vaculín, R.: Data centric BPM and the emerging case management standard: a short survey. In: La Rosa, M., Soffer, P. (eds.) *BPM Workshops 2012. LNBIP*, vol. 132, pp. 24–30. Springer, Heidelberg (2013)
3. Heath, F.T., Hull, R., Vaculin, R.: Barcelona: a design and runtime environment for modeling and execution of artifact-centric business processes. In: *Demo Track in International Conference on Business Process Management 2011* (2011)
4. Boaz, D., Heath, T., Gupta, M., Limonad, L., Sun, Y., Hull, R., Vaculín, R.: The ACSI hub: a data-centric environment for service interoperation. In: *Proceedings of the BPM Demo Sessions 2014 Co-located with the 12th International Conference on Business Process Management (BPM 2014)*. Volume 1295 of *CEUR Workshop Proceedings*, CEUR-WS.org (2014)
5. Baresi, L., Bianculli, D., Ghezzi, C., Guinea, S., Spoletini, P.: Validation of web service compositions. *IET Softw.* **1**(6), 219–232 (2007)
6. Alonso, G., Casati, F., Kuno, H.A., Machiraju, V.: *Web Services - Concepts Architectures and Applications. Data-Centric Systems and Applications*. Springer, Heidelberg (2004)
7. Graf, S., Saidi, H.: Construction of abstract state graphs with PVS. In: Grumberg, O. (ed.) *CAV 1997. LNCS*, vol. 1254, pp. 72–83. Springer, Heidelberg (1997)
8. Belardinelli, F., Lomuscio, A., Patrizi, F.: Verification of deployed artifact systems via data abstraction. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) *Service Oriented Computing. LNCS*, vol. 7084, pp. 142–156. Springer, Heidelberg (2011)

9. Belardinelli, F., Lomuscio, A., Patrizi, F.: An abstraction technique for the verification of artifact-centric systems. In: Proceedings of Principles of Knowledge Representation and Reasoning (KR 2012), pp. 319–328 (2012)
10. Belardinelli, F., Lomuscio, A., Patrizi, F.: Verification of GSM-based artifact-centric systems through finite abstraction. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) Service Oriented Computing. LNCS, vol. 7636, pp. 17–31. Springer, Heidelberg (2012)
11. Bhattacharya, K., Gerede, C.E., Hull, R., Liu, R., Su, J.: Towards formal analysis of artifact-centric business process models. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 288–304. Springer, Heidelberg (2007)
12. Deutsch, A., Hull, R., Patrizi, F., Vianu, V.: Automatic verification of data-centric business processes. In: Proceedings of the 12th International Conference on Database Theory (ICDT 2009), Volume 361 of ACM International Conference Proceeding Series, pp. 252–267. ACM (2009)
13. Belardinelli, F., Lomuscio, A., Patrizi, F.: Verification of agent-based artifact systems. *J. Artif. Intel. Res.* **51**, 333–376 (2014)
14. Gonzalez, P., Griesmayer, A., Lomuscio, A.: Verifying GSM-based business artifacts. In: Proceedings of the IEEE International Conference on Web Services (ICWS 2012), pp. 25–32 (2012)
15. Gonzalez, P., Griesmayer, A., Lomuscio, A.: Model checking GSM-based multi-agent systems. In: Lomuscio, A.R., Nepal, S., Patrizi, F., Benatallah, B., Brandić, I. (eds.) ICSOC 2013. LNCS, vol. 8377, pp. 54–68. Springer, Heidelberg (2014)
16. Shoham, S., Grumberg, O.: 3-valued abstraction: more precision at less cost. *Inf. Computat.* **206**(11), 1313–1333 (2008)
17. Hull, R., et al.: Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events. In: Proceedings of the 5th ACM International Conference on Distributed Event-Based Systems (DEBS 2011) (2011)
18. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. The MIT Press, Cambridge (1995)
19. Chechik, M., Devereux, B., Easterbrook, S., Gurfinkel, A.: Multi-valued symbolic model-checking. *ACM Trans. Softw. Eng. Methodol.* **12**(4), 371–408 (2003)
20. Bozianu, R., Dima, C., Enea, C.: Model-checking an epistemic μ -calculus with synchronous and perfect recall semantics (2013). CoRR [abs/1310.6434](https://arxiv.org/abs/1310.6434)
21. Kozen, D.: Results on the propositional μ -calculus. *Theor. Comput. Sci.* **27**(3), 333–354 (1983)
22. Gonzalez, P., Griesmayer, A., Lomuscio, A.: GSMC: a model checker for GSM (2014). <http://vas.doc.ic.ac.uk/software/extensions/>
23. Somenzi, F.: CUDD: CU decision diagram package release 2.5.0 (2012). <http://vlsi.colorado.edu/~fabio/CUDD/>
24. Barrett, C., Tinelli, C.: CVC4 version 1.2 (2013). <http://cvc4.cs.nyu.edu/web/>
25. Huth, M., Ryan, M.: Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge University Press, Cambridge (2004)
26. Toribio Gomez, D., Murphy-O'Connor, C., De Leenheer, P., Malarme, P.: D5.5 deployment and evaluation of pilots using final ACSI hub system results and evaluation. Project deliverable, The ACSI Project (EU FP7-ICT-257593) (2013)
27. Group, O.M.: Case management model and notation, version 1.0. Technical report (2014)

Mining and Querying Process Change Information Based on Change Trees

Georg Kaes^(✉) and Stefanie Rinderle-Ma

Faculty of Computer Science, University of Vienna, Vienna, Austria
{georg.kaes,stefanie.rinderle-ma}@univie.ac.at

Abstract. Analyzing process change logs provides valuable information about the evolution of process instances. This information can be used to support responsible users in planning and executing future changes. Change mining results in a change process, which represents the dependencies between process changes mined from the change log. However, when it comes to highly adaptive process settings, multiple limitations of the change process representation can be found, i.e., based on change processes it is not possible to provide answers to important analysis questions such as ‘How many instances have evolved in a similar way?’ or ‘Which changes have occurred following a particular change?’. In this paper, change trees and n-gram change trees are introduced to serve as a basis to analyze changes in highly adaptive process instances. Moreover, algorithms for discovering change trees and n-gram change trees from change logs are presented. The applicability of the approach is evaluated based on a systematic comparison with change mining, a proof-of-concept implementation and by analyzing real-world data.

1 Introduction

Process change and evolution is a key concern in many application domains such as care [1], manufacturing [1, 2], logistics [3], and health care [4]. The management of change information in business processes is a relevant challenge for flexible process aware information systems (PAIS) [4]. Change logs are a central asset to log, manage and understand how a process (instance) evolves over time [5]. Approaches such as [6, 7] advocate the exploitation of knowledge on previous changes for supporting users in applying future changes. Whereas [6] analyzes user annotations, [7] presents change mining to discover change processes from change logs. The resulting change processes visualize possible orderings of changes and contain information about possible dependencies between changes. However, as it will be shown, change processes are not suitable for answering the following analysis questions:

Q1: Which process instances have evolved in a similar way?

Q2: Which process instances have evolved in a similar way after a certain change sequence?

Both questions are relevant in practical settings. Analyzing which process instances have evolved in a similar way can be used as a starting point for predicting changes which may become necessary in the future. In the care domain, for example, Q1 can be used to analyze patients' treatment histories in order to predict future changes. Imagine two patients which have received the same treatments over a course of time, whereby the first patient's treatment plan is already further developed than the treatment plan of the second patient. When the question arises which changes may be necessary for the second patient's treatment plan in the future, the change information from the first patient's treatment plan can be used as a starting point.

Q2 provides more focus by asking for, e.g., the development of a treatment after a certain therapy was applied. This information can be used as a basis for analyzing what usually happens after a certain set of changes has been applied. Imagine a therapy in the nursing home setting which requires the nurses to conduct multiple other therapies afterwards, maybe because of possible complications. The question of interest is now, how many treatment plans have evolved in which way after this certain therapy has been conducted. This information can be used to further enhance the planning and conduction of therapies.

The question is how change information of process instances can be represented such that Q1 and Q2 can be sufficiently analyzed. As both Q1 and Q2 refer to the process instance level and as we assume that changes might be applied multiple times (think, for example, of steps such as physical therapy that might become necessary multiple times), a suitable representation must meet the following requirements:

R1: ability to deal with multiple occurrences of (change) instances (Q1, Q2)

R2: ability to deal with multiple occurrences of changes (Q1, Q2)

R3: ability to detect change sequences that follow a certain change pattern (Q2)

Existing approaches such as change mining do not fully meet these requirements. Hence, this paper introduces two new representations for information stored in change logs¹. At first, the change tree is defined in Sect. 3 as a basis to meet requirements R1 and R2. In addition to the formal definition an algorithm is presented that mines change trees from change logs. Section 4 introduces the n-gram change tree as a further development of the change tree meeting requirement R3. Based on representing change sequences as n-grams the n-gram change tree offers a projection on those parts of the change instances that evolved after the occurrence of the n-gram. It is shown how n-gram change trees can be constructed from change logs. The feasibility of the change tree and n-gram change tree is evaluated in several ways. Section 5 systematically compares change tree and n-gram change tree to other representations such as change processes and graphs. A proof-of-concept implementation as well as an application of the concepts to a real-world data set from the BPI 2014 challenge is presented in Sect. 6. Section 7 discusses related approaches and Sect. 8 concludes the paper and gives an overview over future work.

¹ Fundamental definitions of changes and change logs are presented in Sect. 2.

Overall, change tree and n-gram change tree are novel change log representations that meet requirements R1, R2, and R3 and hence enable the in-depth analysis of highly dynamic process applications.

2 Change Log Definitions

The following definitions of changes and change logs are based on the recommendations from literature. A *change* Δ is defined according to [5] as $\Delta := (type, subject, paramList, S)$

Typical change *types* as summarized in the change patterns collection [8] are INSERT, DELETE, or MOVE. *Subject* refers to the task or activity to be, e.g., inserted or deleted. The *paramList* specifies, for example, the context of a change. Finally, *S* is the (instance) schema the change is applied to. Change $\Delta = (\text{INSERT}, A, \langle B, C \rangle, S)$ inserts activity A between activities B and C in instance schema S.

Making a simplification to [5], a *change log* is defined as

$$cL := \langle \Delta_1, \dots, \Delta_n \rangle \quad (1)$$

Assume the following change log

$$cL = \langle \Delta_1 = (\text{INSERT}, A, \langle \text{Therapy Fragment C}, \text{End} \rangle, S_{I1}), \\ \Delta_1 = (\text{INSERT}, A, \langle \text{Therapy Fragment C}, \text{End} \rangle, S_{I2}), \\ \Delta_2 = (\text{INSERT}, B, \langle \text{Therapy Fragment C}, \text{End} \rangle, S_{I2}) \rangle.$$

The implicit assumption of an ordering suggests that two times an activity A was inserted between Therapy Fragment C and End, followed by an insertion of activity B between Therapy Fragment C and End.

An MXML-based format for change logs was proposed in the context of change mining [9]. Listing 1.1 shows the MXML representation for cL.

Listing 1.1. Fragment for Change Log Example in MXML

```
<WorkflowLog ...>
  <Process id="OR">
    <ProcessInstance id="1">
      <AuditTrailEntry>
        <Data>
          <Attribute name="CHANGE.postset">End</Attribute>
          <Attribute name="CHANGE.type">INSERT</Attribute>
          <Attribute name="CHANGE.subject">A</Attribute>
          <Attribute name="CHANGE.rationale">Therapy Fragment A required</Attribute>
          <Attribute name="CHANGE.preset">Therapy Fragment C</Attribute>
        </Data>
        <WorkflowModelElement>INSERT.A</WorkflowModelElement>
        <EventType>complete</EventType>
        <Originator>Dr. Ford</Originator>
      </AuditTrailEntry>
    </ProcessInstance>
    <ProcessInstance id="2">
      <AuditTrailEntry>
        <Data>
          <Attribute name="CHANGE.postset">End</Attribute>
          <Attribute name="CHANGE.type">INSERT</Attribute>
          <Attribute name="CHANGE.subject">A</Attribute>
```



```

    <Attribute name="CHANGE.rationale">Therapy Fragment A required</Attribute>
    <Attribute name="CHANGE.preset">Therapy Fragment C</Attribute>
  </Data>
  <WorkflowModelElement>INSERT.A</WorkflowModelElement>
  <EventType>complete</EventType>
  <Originator>Dr. Ford</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <Data>
    <Attribute name="CHANGE.postset">End</Attribute>
    <Attribute name="CHANGE.type">INSERT</Attribute>
    <Attribute name="CHANGE.subject">B</Attribute>
    <Attribute name="CHANGE.rationale">Therapy Fragment B required</Attribute>
    <Attribute name="CHANGE.preset">Therapy Fragment C</Attribute>
  </Data>
  <WorkflowModelElement>INSERT.B</WorkflowModelElement>
  <EventType>complete</EventType>
  <Originator>Dr. Dent</Originator>
</AuditTrailEntry>
</ProcessInstance>
</Process>
</WorkflowLog>

```

Comparing Eq. 1 and the MXML-based representation in Listing 1.1, the latter collects changes at the instance level whereas the former refers to the schema level. In fact, change mining aggregates the instance-specific change information into an analysis model, i.e., the change process. Figure 1 shows the result of applying change mining to the log in Listing 1.1 as produced by the Change Miner plugin of the ProM 5.2 framework².

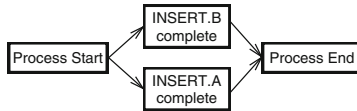


Fig. 1. Change process resulting from change mining (using ProM 5.2)

For the considerations of this paper, a process instance is characterized by the changes that have been applied to the instance³. Hence, in the following we only refer to *change instances* (*instances* for short).

Definition 1 (Change Instance, Change Log). Let \mathcal{I} be a set of process instances and \mathcal{C} be a set of changes. For each $I \in \mathcal{I}$, S_I denotes the instance schema, i.e., the schema the instance is currently running on, and $\Delta_1^I, \dots, \Delta_n^I$ reflect the changes applied to I (S_I) so far, $\Delta_j^I \in \mathcal{C}, j = 1, \dots, n$. Assume that Δ_i^I was applied before Δ_j^I if $i < j, i, j \in \{1, \dots, n\}$. Then a change instance I is defined as follows:

$$I : \Delta_1^I \rightarrow \Delta_2^I \rightarrow \dots \rightarrow \Delta_n^I$$

A change log cL represents a collection of change instances.

² <http://www.promtools.org/doku.php?id=prom52>.

³ i.e., any other information such as (instance) schema or execution state will be considered in future work.

For Listing 1.1 the change log cL with the associated change instances turns out as follows:

- I1: Δ_1
 I2: $\Delta_1 \rightarrow \Delta_2$.

3 Change Trees

In order to be able to answer Q1 and meet requirements R1 and R2 (cf. Sect. 1), a representation for change information shall represent the chronological order of changes made to all instances in one aggregated view. Definition 2 presents the change tree as a representation for instance change information. Intuitively the change tree represents each of the change instances in a change log along with the number of its occurrences along paths from the root to the leafs.

Definition 2 (Change Tree). *Let cL be a change log and C be the changes contained in cL . Then change tree T is defined as a rooted multiway tree $T := (r, V, E)$ with*

1. $r := \emptyset$ is the unique root node
2. $V \subseteq C \times \mathbb{N}_0$
3. \forall leaf nodes $v = (\Delta, n) \in V: n > 0$
4. \forall paths p from root r to node $v = (\Delta, n) \in V$ with $n > 0$: p corresponds to n change instances in cL .

To complete Definition 2, Algorithm 1 sets out how new changes are added to a change tree. Deletion or reorganization on change trees do not become necessary since removing change Δ from the change tree is considered as adding a “compensating” change Δ' .

Algorithm 1. Adding a Change to a Change Tree

Input:

- change Δ (applied to instance I)
- change tree CT (which contains instance I)

```

1 Begin
2 currentnode = root of CT
3 for  $i = 0; i < I.length; i++$  do
4   | currentnode = child node where I is stored
5 if there is a child of currentnode containing  $\Delta$  then
6   | Decrement count of currentnode
7   | Go to the node containing  $\Delta$ 
8   | Increment the count of this node
9 else
10  | Decrement count of currentnode
11  | Create a new child node for currentnode containing  $\Delta$ 
12  | Set the count of this node to 1
13 End

```

Let us illustrate the concept of the change tree along the example depicted in Fig. 2. Figure 2 shows a change log (left side) and its representation as a change tree (right side)⁴. Change instance I1 for example consists of two consecutive applications of change Δ_1 (*R2*: multiple occurrence of changes). Starting from the root node and going towards *Leaf 2* in the change tree, the change instance can be reproduced. The number *1x* in *Leaf 2* indicates that this change instance, i.e., $\Delta_1 \rightarrow \Delta_1$, occurred once in the change log. With this, the change tree offers the possibility to count the number of multiple instance occurrences.

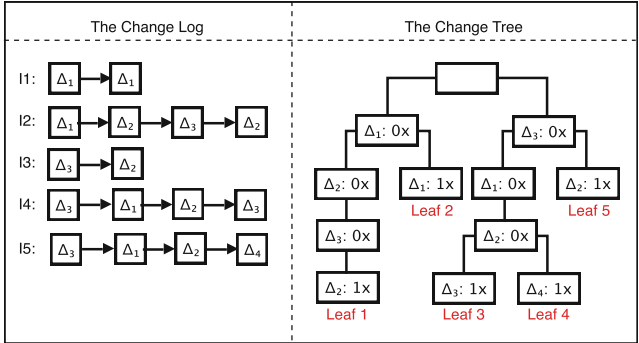


Fig. 2. Change log and corresponding change tree

In order to illustrate how a change is added to a change tree (cf. Algorithm 1) consider Fig. 3 which depicts two scenarios based on a change instance $I : \Delta_1$ and a given change tree (left side). In a first scenario change Δ_1 is again applied to I such that I is updated to $I : \Delta_1 \rightarrow \Delta_1$. As a first step I is located in the change tree (lines 2–4 in Algorithm 1). As a second step we look if the currently applied change has already been applied to some other change instance I' with $I = I'$ for I before applying Δ_1 the second time (lines 5–9). If such I' exists, the change instance is moved further up the change tree to the next level. This is the case for I in scenario 1. Now assume in a second scenario that not Δ_1 has been applied a second time, but Δ_3 instead. In this case, no I' exists with $I = I'$ (before the new change). Thus a new branch in the change tree is created and the change instance is moved there (change Δ_3 , second and third pane).

Overall, the change tree representation covers all instances in the log as well as maintains the number of occurrences of change instances (*R1*) and change occurrences (*R2*). With respect to question *Q1* as set out in the introduction, the change tree offers the possibility to determine how many instances have evolved in which way. This information can be important for predicting and planning future changes.

Algorithm 2 sets out how a change tree can be mined from a change log. The algorithm starts with the first set of changes which has been applied to

⁴ For illustration reasons the change tree is annotated with the number of leaves.

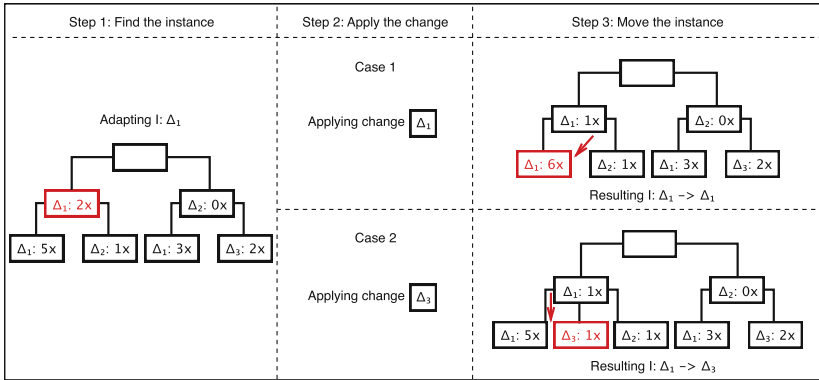


Fig. 3. Adding a change to a change tree

the process instances in the change log (line 30). For instances I1 and I2 from Fig. 2 this is Δ_1 , for I3, I4 and I5 this is Δ_3 (the first row of changes). For each instance in the given set of instances (for the root node this are all instances in the change log) the change at the current position is checked. If a node containing this change does not already exist a new node is generated. For the first iteration two nodes containing Δ_1 and Δ_3 are generated. Next we check if the change we just created the node for is the last change for this process instance or not. If it is, we increment the counter for this node. This means that the instance is finished, we have reached leaf level and we have found one more instance with the given set of changes. If it is not the last instance we add the current instance to the set of instances for this node which will be traversed in later iterations. The set of instances is used to generate the child nodes for our current node, and since there is a following change, there will be a child node. Now we recursively reuse this function to generate all child nodes for each node in our set of nodes. After the first iteration this means that the function is called for the nodes Δ_1 (with instances I1 and I2) and Δ_3 (with instances I3, I4 and I5). Note that the iterator is incremented, thus analyzing the second level of changes (instances I1, I4 and I5: Δ_1 , instances I2 and I3: Δ_2).

4 n-Gram Change Trees

In order to answer Q1 (cf. Sect. 1), the change tree represents the chronological order of changes made to all instances in one aggregated view. In order to answer Q2, in addition, all occurrences of a specific change sequence must be detected and “consolidated” in the resulting representation in order to analyze what happened after the change sequence to the process instances of interest.

A change instance as defined in Definition 1 can also be understood as a string and a change sequence as substring. Thus, the problem can be considered as a transformation of the problem of *n-gram models* in language processing where two strings are defined as equivalent “if they end in the same $n - 1$ words” [10].

Algorithm 2. Create a change tree from a change log

```

Input: Change log  $cL$ 
1 //parent is the parent node for the children which are inserted
2 //set-of-instances is the set of instances which contains data for child nodes of the
  current parent node
3 //iterator is the number of the change in the change log instance - iterator=2 means the
  2nd change applied to the instance
4 function create-children(parent,set-of-instances,iterator)
5   set-of-nodes = new Array
6   foreach instance in set-of-instances do
7     if there is no child node with the current change for this parent then
8       node = create-node(instanceiterator,parent)
9       set-of-nodes.add(node)
10    else
11      node = the child node of the parent containing the current change
12    if instanceiterator+1==empty then
13      node.count++
14    else
15      node.nextinstances.add(instance)
16  foreach node in set-of-nodes do
17    create-children(node,node.nextinstances,iterator+1)
19  return ;

20 function create-node(label,parent)
21   node.label = label
22   node.count = 0
23   node.nextinstances = new Array
24   node.children = new Array
25   if parent then
26     parent.children.add(node)
27   return node

28 Begin
29 root = create-node(null,null);
30 create-children(root, change log, 0);

```

Hence, the n -gram in the change setting will reflect the change pattern (substring) of interest and we will determine the change sequences following the change pattern for each of its occurrences in the log.

Assume that we are interested in the n -gram $\Delta_1 \rightarrow \Delta_2$ and consider the example depicted in Fig. 4. For instance I2, the n -gram can be found at the beginning whereas for instances I4 and I5 the n -gram occurs after Δ_3 . This results in a change tree where the required sequence of changes can be found in two different branches of the change tree (red nodes in the middle pane).

To see all changes following n -gram $\Delta_1 \rightarrow \Delta_2$, the two subtrees containing the n -gram have to be combined, i.e. restructuring the change tree becomes necessary. This can be achieved by using a suffix tree [11] or, more precisely, a generalized suffix tree which contains more than one string. A suffix tree represents all suffixes for a given string. Suffix trees are commonly used for pattern matching in various scenarios, from string matching to finding common motifs

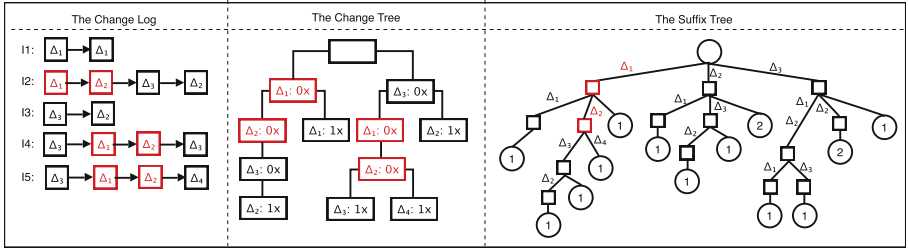


Fig. 4. Development of the n-gram change tree

in DNA sequences [12]. In contrast to simple implementations of the suffix tree the algorithm proposed by Ukkonen [13] works in linear time $O(n)$, where n represents the length of the string.

In the generalized suffix tree in Fig. 4 the leaf node of path $Root \rightarrow \Delta_2 \rightarrow 2$ means that two instances have change Δ_2 as their suffix. There are also two instances which end with the changes $\Delta_3 \rightarrow \Delta_2$ (instance I2 and I3), but only one instance which ends with $\Delta_2 \rightarrow \Delta_3$ (instance I4).

As depicted in Fig. 4 the suffix tree is constructed over the suffixes for all strings which can be found in the set of change logs for process instances I1 to I5. For n-gram $\Delta_1 \rightarrow \Delta_2$ we can now easily see the combined subtrees since all suffixes which start with the sequence can be found directly at the root node (marked red in Fig. 4, third pane). This information can be represented as the n-gram change tree, depicted in the second pane of Fig. 5. The n-gram change tree contains the n-gram in its root node. The suffixes of the n-gram, i.e., $\Delta_3 \rightarrow \Delta_2$, Δ_3 , and Δ_4 have produced two paths from root to leaves. Specifically, $\Delta_3 \rightarrow \Delta_2$ and Δ_3 are aggregated into one such path.

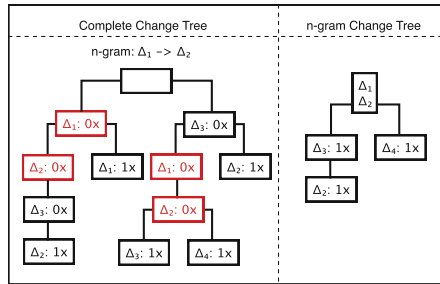


Fig. 5. All occurrences of the n-gram in the change tree and the n-gram change tree

The n-gram change tree can be defined similarly to the change tree with some modifications as set out in Definition 3.

Definition 3 (n-Gram Change Tree). Let cL be a change log and C be the changes contained in cL . Then n-gram change tree nT is defined as a change tree (cf. Definition 2) where the following conditions are different:

1* $r := \langle \Delta_1, \dots, \Delta_l \rangle$ where $\Delta_i \in C$, $i = 1, \dots, l$. is the unique root node
 4* \forall paths p from root r to node $v = (\Delta, n) \in V$ with $n > 0$: p corresponds to a projection of n change instances in cL starting from $\langle \Delta_1, \dots, \Delta_l \rangle$ as defined in the root node.

Algorithm 1 also works for n-gram change trees with the only difference that the change is only added to the n-gram change tree if it has been applied after the respective n-gram has been applied to an instance.

Preparing the Change Logs for Queries: In a complex setting such as the nursing domain, often multiple changes are made, which are not necessarily interacting with each other. For example, a patient has problems with his right knee, but at the same time he is on a special diet because of certain allergies. In these cases, we have to **trim the aforementioned data structure, so only the relevant changes are analyzed**. Based on these trimmed change logs, we can analyze the remaining logs with either the change tree or the n-gram change tree.

5 Comparison with Other Representations

This section compares the change tree and the n-gram change tree to other representations such as the change process [9] and graph-based structures. The basis for the evaluation is Listing 1.1 with an extension by multiple instances⁵. Recall for the following examples that

$\langle \Delta_1 = (\text{INSERT}, A, \{\text{Therapy Fragment C}, \text{End}\}, S) \text{ and}$
 $\Delta_2 = (\text{INSERT}, B, \{\text{Therapy Fragment C}, \text{End}\}, S) \rangle$.

First consider the scenario depicted in Table 1. On the left side the change log is depicted, in the middle the resulting change process after applying change mining, and on the right side the change tree. Note that the resulting change process as for example shown in Fig. 1 has been transformed into a Petri Net in order to reason about the semantics of splits.

For the scenario in Table 1 the change process does not convey the information that for 9 instances change Δ_1 has been applied while change Δ_2 has only been applied for two instances. Moreover, based on the OR-split, it is not possible to see that for 3 instances first Δ_1 and then Δ_2 has been applied while for one scenario the reverse order of change occurred. The reason for this limitation is that change mining abstracts from the number of instance occurrences. In contrast to this the change tree reflects multiple occurrences of change instances, thus fulfilling requirement *RI*. All possible combinations of changes as they can be found in the change logs can be easily detected and interpreted. For example it can be concluded that the probability of having change Δ_1 is nine times the probability of change Δ_2 .

In the second scenario (cf. Table 2) the change process cannot correctly reflect the difference between the scenario where only change Δ_1 has been applied, and the two others where Δ_1 respectively Δ_2 followed Δ_1 . The multiple occurrence

⁵ The logs can be found on <http://cs.univie.ac.at/project/apes>.

Table 1. Multiple instance occurrences

Change Log	Change Process	Change Tree
I01-I09: Δ_1 I10: Δ_2 I11-I13: $\Delta_1 \rightarrow \Delta_2$ I14: $\Delta_2 \rightarrow \Delta_1$		

of the same change cannot be reflected correctly in the change process. The change tree in Table 2 removes the inaccuracy of the change process regarding the occurrence of multiple changes at different points in time. We can now easily see that in the given situation for each case change Δ_1 has been applied first. In half of the cases afterwards change Δ_1 has been applied again - in the other half change Δ_2 was used.

Table 2. Distinction of change occurrences

Change Log	Change Process	Change Tree
I1: Δ_1 I2: $\Delta_1 \rightarrow \Delta_1$ I3: $\Delta_1 \rightarrow \Delta_2$		

The problem that multiple occurrences of the same change cannot be correctly reflected in the change process is aggravated in the third scenario shown in Table 3. Here, different process scenarios still produce the same change process. The fact that in two cases Δ_1 has been applied only once are not reflected in the

Table 3. Multiple change occurrences

Change Log	Change Process	Change Trees
I1: Δ_1 I2: Δ_1 I3: $\Delta_1 \rightarrow \Delta_1$ or I1: $\Delta_1 \rightarrow \Delta_1$ I2: $\Delta_1 \rightarrow \Delta_1$		

resulting change process. The corresponding change trees clearly show a distinction between the instances where Δ_1 has been applied once or multiple times, thus fulfilling requirement *R2*.

Representing change logs as graph structure instead of a tree would lead to information loss and is thus not a viable solution. Consider the tree and the graph representation in Fig. 6. The change tree on the left has been compressed to a graph structure which requires fewer nodes. Each edge in the graph contains information about how many instances have evolved in the respective direction, so it can be seen that after Δ_1 on the first level four instances have evolved to Δ_3 (and possibly beyond this point). As can be discovered easily in the change tree, only one instance stopped its evolution here at Δ_3 while 2 instances evolved further to Δ_1 and one instance evolved to Δ_2 . However, this information is lost in the graph representation since the two Δ_3 nodes would be merged.

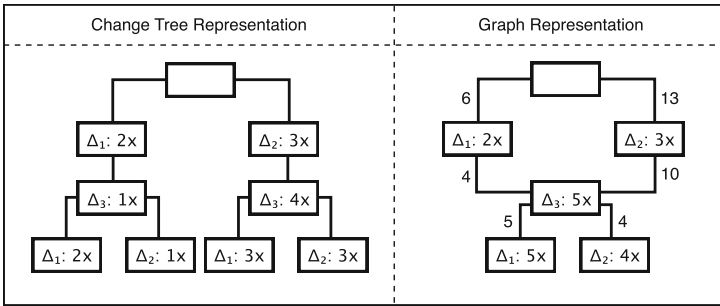


Fig. 6. Comparing tree- and graph based models

6 Proof-of-Concept and Real-World Example

In order to provide experts with the possibility to use the change tree and the n-gram change tree for their own projects, we implemented it as a ProM plugin⁶ Based on MXML and XES log files one can build a change tree, select an n-gram and generate the n-gram change tree (cf. Fig. 7).

During the Business Process Intelligence (BPI) Challenge real world process logs are analyzed from various points of view. The BPI Challenge 2014⁷ was based on data from different processes of Rabobank Group ICT. When analyzing the log files we found that one of these processes is suited to be analyzed by the change tree since its log provides a set of activities which describe what has happened to a specific item in order to solve some problem. These activities are

⁶ The current version of the plugin is available as a nightly build at <http://www.promtools.org/prom6/nightly/>.

⁷ <http://www.win.tue.nl/bpi/2014/challenge>, doi:10.4121/uuid:c3e5d162-0cfd-4bb0-bd82-af5268819c35.

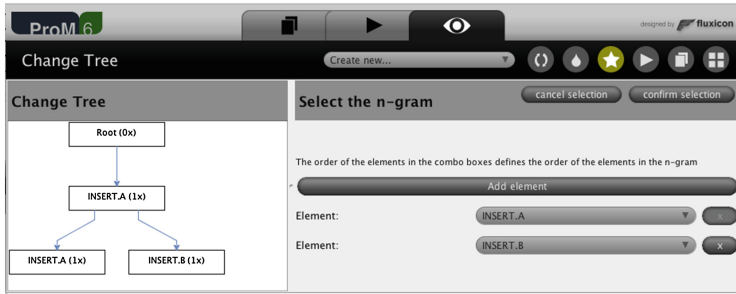


Fig. 7. Implementation of the change tree as a ProM plugin

the basic building blocks for our change tree: Each time a new activity is planned for a specific item, the process of this item changes. Thus, the change tree can be used to mine change information from these log files.

By mining process logs with the change tree we want to find information about what has usually happened after a certain change or a set of changes. For example we found multiple repeating process steps after including “Standard Change Type 88” (SCT 88) into the process. This is reflected by the change tree depicted in Fig. 8. Specifically, the tree is a 1-gram change tree with 1-gram “Standard Change Type 88”.

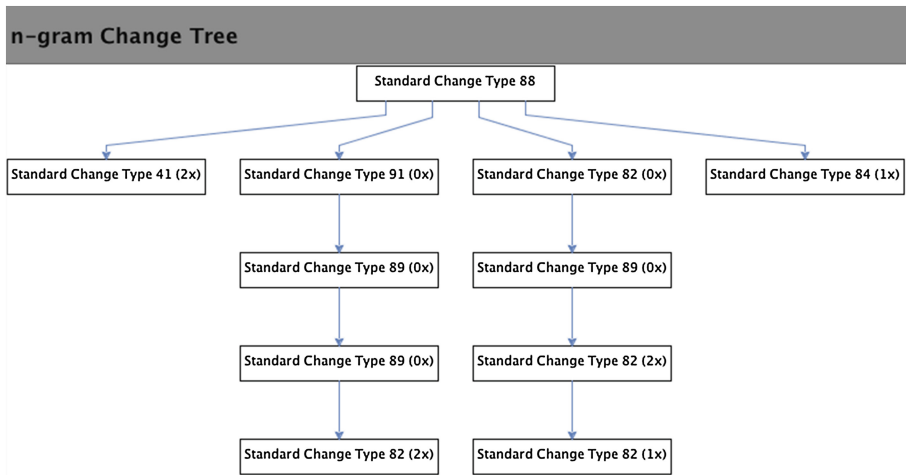


Fig. 8. Process steps following Standard Change Type 88

From the 1-gram change tree it can be concluded that change “Standard Change Type 88” was followed in most cases by a change sequence containing “Standard Change Type 82” or “Standard Change Type 41”. Since the provided data lacks context information it is not possible to deduce any semantical information from the generated change tree. If more information was available one

could predict certain requirements for future tasks based on the change tree. Think for example of knowledge about resources required for a specific task, such as requiring a technical specialist for executing “Standard Change Type 82”. Using the n-gram change tree as a resource planning instrument it can be predicted that after executing change “Standard Change Type 88”, with a certain probability “Standard Change Type 82” will become necessary as well and a technical specialist will be required (even if not required for “Standard Change Type 88” in the first place).

7 Related Work

Change of process instances and process schemas have been analyzed from various vantage points.

[7,9] describe the generation of a change process based on change mining. This method of analyzing change logs provides valuable information, especially for process instances which are based on a common schema, for example, on how to improve the process schema itself. However, as shown in this paper, change mining is not suited to reflect multiple change instances and multiple occurrences of changes. Moreover, searching for change patterns and their subsequent changes is not supported.

[6] aims at supporting users when applying change operations as well. For this, users can annotate changes with explanations and the systems exploits the changes by their frequencies together with the annotations in order to suggest changes to users. Change trees and n-gram change trees do not consider additional change annotations. However, in the presence of such annotations, [6] can provide complementary information to users.

[14] focuses on analyzing process variants, which are derived from a common process schema. The authors’ goal is to find the process schema, where the smallest set of changes has to be applied to in order to obtain the schema of the individual process instances. In a first scenario, the existence of a reference process schema is assumed. This reference schema is adapted such that as few as possible additional changes will be necessary to reflect the process instances. In the second part, a process schema is generated solely based on the process instances and their changes. This approach does not construct analysis models from change logs, but it can serve as valuable complement to the approach presented in this paper. For example, one could find the change with the smallest set of required changes and use it as a basis for future changes.

Changes in the process instances schema cannot only be derived from the change log, but also from the event log. [15] presents methods to detect sudden changes in the process schema solely based on event log entries. For analyzing the effects of a change, such a system would also be of interest: Imagine a doctor who adds a new therapy to a patient where drug X has to be applied each week. It is generally known that drug X cannot be given at the same time as drug Y, which the patient currently receives. However for some reason the administration of drug Y has not been removed from the patient’s therapy plan,

and the next time the doctor sees that he should administer drug Y he just skips the corresponding process task. Such effects of changes, which cannot be detected based on the change tree could be analyzed with such a system.

8 Conclusion and Future Work

This paper introduced change trees and n-gram changes together with the associated mining algorithms in order to discover analysis models from change logs that support users in deciding on future change application based on previously applied changes. The benefit of change trees – specifically when compared to existing change mining results – is that they reflect multiple change instances and multiple change occurrences. Both are characteristic to highly adaptive process scenarios. In addition, n-gram change trees enable answers to questions such as ‘*which changes happened after the occurrence of a certain change pattern?*’. This can be very interesting for users, as they do not have to search possibly complex change tree structures containing all the information in a change log, but a “projection” of the trees to the information of interest. Change trees and n-gram change trees have been evaluated in several ways: we compared them to existing change mining techniques and graph based methods, provided a technical implementation and an application to a real-world log.

Change trees reflect the structural aspect of change logs. Specifically, change instances and patterns are only considered as equal if they contain exactly the same changes. For practical settings it might be also of interest to consider ‘similar’ change sequences and patterns, i.e., go from a structural point of view to a more semantic one. This also might necessitate the inclusion of additional information such as process instance execution state or other instance parameters.

Data mining techniques such as Generalized Sequential Patterns might be useful to narrow down the number of possible change sequences to those which are statistically significant. Especially in change logs where a large number of different process changes appear, such an approach can significantly increase the usability of the change tree. Additionally, the analysis of scalability and efficiency when it comes to very large process logs is an interesting topic for future work.

References

1. Kaes, G., Rinderle-Ma, S., Vigne, R., Mangler, J.: Flexibility requirements in real-world process scenarios and prototypical realization in the care domain. In: Meersman, R., et al. (eds.) OTM Workshops. LNCS, vol. 8842, pp. 55–64. Springer, Heidelberg (2014)
2. Schulte, S., Schuller, D., Steinmetz, R., Abels, S.: Plug-and-play virtual factories. *IEEE Internet Comput.* **16**, 78–82 (2012)
3. Bassil, S., Keller, R.K., Kropf, P.G.: A workflow-oriented system architecture for the management of container transportation. In: Desel, J., Pernici, B., Weske, M. (eds.) BPM 2004. LNCS, vol. 3080, pp. 116–131. Springer, Heidelberg (2004)
4. Reichert, M., Weber, B.: *Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies*. Springer, Heidelberg (2012)

5. Rinderle, S., Reichert, M., Jurisch, M., Kreher, U.: On representing, purging, and utilizing change logs in process management systems. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 241–256. Springer, Heidelberg (2006)
6. Weber, B., Reichert, M., Rinderle-Ma, S., Wild, W.: Providing integrated life cycle support in process-aware information systems. *Int. J. Coop. Inf. Syst.* **18**, 115–165 (2009)
7. Günther, C., Rinderle-Ma, S., Reichert, M., van der Aalst, W.: Using process mining to learn from process changes in evolutionary systems. *Int. J. Bus. Process Integr. Manag.* **3**, 61–78 (2008)
8. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features - enhancing flexibility in process-aware information systems. *Data Knowl. Eng.* **66**, 438–466 (2008)
9. Günther, C.W., Rinderle, S., Reichert, M., van der Aalst, W.: Change mining in adaptive process management systems. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 309–326. Springer, Heidelberg (2006)
10. Brown, P., Desouza, P., Mercer, R., Della Pietra, V., Lai, J.: Class-based n-gram models of natural language. *Comput. Linguist.* **18**, 467–479 (1992)
11. McCreight, E.M.: A space-economical suffix tree construction algorithm. *J. ACM* **23**, 262–272 (1976)
12. Sagot, M.-F.: Spelling approximate repeated or common motifs using a suffix tree. In: Lucchesi, C.L., Moura, A.V. (eds.) LATIN 1998. LNCS, vol. 1380, pp. 374–390. Springer, Heidelberg (1998)
13. Ukkonen, E.: On-line construction of suffix trees. *Algorithmica* **14**, 249–260 (1995)
14. Li, C., Reichert, M., Wombacher, A.: Mining business process variants: Challenges, scenarios, algorithms. *DKE* **70**, 409–434 (2011)
15. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P., Žliobaitė, I., Pechenizkiy, M.: Handling concept drift in process mining. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 391–405. Springer, Heidelberg (2011)

Property Preservation in Adaptive Case Management

Rik Eshuis¹ (✉), Richard Hull², and Mengfei Yi¹

¹ School of Industrial Engineering, Eindhoven University of Technology,
Eindhoven, The Netherlands

`H.Eshuis@tue.nl`, `m.yi@student.tue.nl`

² IBM T. J. Watson Research Center, Yorktown Heights, New York, USA
`hull@us.ibm.com`

Abstract. Adaptive Case Management (ACM) has emerged as a key BPM technology for supporting unstructured business process, and has been used to support flexible services orchestration. A key problem in ACM is that case schemas need to be changed to best fit the case at hand. Such changes are ad-hoc, and may result in schemas that do not reflect the intended logic or properties. This paper presents a formal approach for reasoning about which properties of a case schema are preserved after a modification, and describes change operations that are guaranteed to preserve certain properties. The Case Management model used here is a variant of the Guard-Stage-Milestone model for declarative business artifacts. Applicability is illustrated using a real-life example.

1 Introduction

Case management has been introduced to support knowledge intensive business processes, which are organized around data artifacts [8, 22, 25]. Case management often needs to support flexible business processes that are performed by knowledge workers. So case management schemas must be easy to change. Adaptive Case Management (ACM) has been proposed as umbrella term for flexible case management [20]. Case Management has been applied in many knowledge-worker driven application areas, including fraud detection, healthcare, education, and social work, and has also been used as a basis to support flexible services orchestration to enable collaboration between enterprises (e.g., [15, 17]).

Designing case management models is hard. The presence of business rules may make it difficult to assess and predict the behavior specified in a case management model or schema. However, changing case management schemas is even harder. Unwanted behavior such as logical errors can be easily introduced by changing a case management model. More generally, a change could have undesirable side effects. Therefore, certain user-defined properties should be preserved in the changed schema.

This paper studies conditions under which case management schemas can be changed while preserving specified properties. We use the Guard-Stage-Milestone (GSM) model; GSM schemas declaratively specify life-cycles of business artifacts.

The meta-model underlying the OMG standard Case Management Model and Notation (CMMN) [3] is based on GSM. In this paper we use a restricted variant of the GSM model, called Fully Acyclic GSM, to enable a focus on the key ideas and the development of informative and useful results. We leave for future research the generalization of the approach to richer variants of GSM.

The paper makes three fundamental contributions. First, we develop a precise definition for testing the preservation of properties. This is based on the notion of *conditional emulatability*, which allows to specify a condition under which executions of one GSM schema can be imitated by executions of a second GSM schema. Second, we develop a general-purpose “*Lifting Lemma*”. Speaking intuitively, this provides a mechanism for isolating changes to a “local area” in a GSM schema. And third, we use the Lifting Lemma to show how *key change operations* can be defined so as to guarantee the preservation of certain properties. Importantly, the theoretical work is motivated by examples arising in a real-world application.

The remainder of this paper is structured as follows. Section 2 introduces the problem of changing GSM schemas based on a real-world example, and illustrates change operations that preserve specified properties. Section 3 formally introduces the GSM model used in this paper. Section 4 develops the Lifting Lemma, and Sect. 5 illustrates applications of the Lifting Lemma by defining general-purpose change operations that preserve selected properties. Section 6 describes related work, and Sect. 7 offers brief conclusions.

Due to space limitations, the presentation here is terse. To improve readability, several of the definitions and some of the results are presented in an informal style. A technical report [10] contains more details on the results.

2 Motivation

To introduce the problem of variability, we consider an example based on a real-world process from an international technology company, which has offices in different geographic regions of the world. In the process, business criteria for partner contracts are assessed. Each region has its own flavor of the process.

Example 2.1. The base process, called here BCA^{base} , is used for the main region and has the following activities (see Fig. 1). First, data is gathered needed to perform the assessment. Next, two activities are performed in parallel as a pre-check. The credit is checked to ensure that the credit limit of the partner is still valid. In parallel, the past performance of the partner is evaluated and checked. If both checks are successful, the pre-check succeeds and a detailed check is performed, which may either succeed or fail. If the pre-check has succeeded within three weeks, a bonus is paid to the team managing the deal.

Figure 1 shows the lifecycle part of the GSM schema for this process. The lifecycle contains stages (rounded rectangles) which represent the business activities (in this paper, these are essentially (atomic) tasks that are not explicitly modeled within the GSM schema). Guards (diamonds) specify under which condition work in a stage is launched. Milestones (circles) represent business objectives

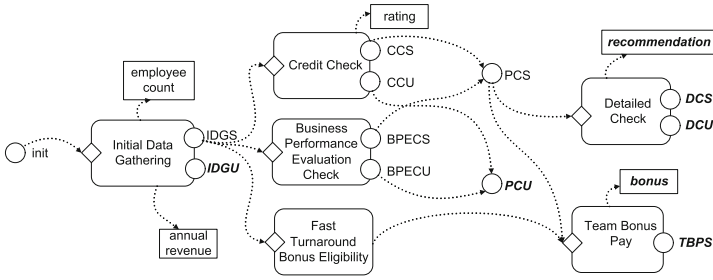


Fig. 1. Main business criteria assessment process (BCA^{base})

Table 1. Stages and guards for BCA^{base} in Fig. 1

Stage	Guard
Initial Data Gathering	init
Credit Check	IDGS
Business Performance Evaluation Check	IDGS
Detailed Check	PCS
Fast Turnaround Business Eligibility	IDGS
Team Bonus Pay	C:Fast Turnaround Business Eligibility \wedge fast_turnaround \wedge PCS

that are achieved by stages to which they are attached or by important events. Guards and milestones have sentries (business rules) that specify when they are executed; these are shown in Tables 1 and 2. Sentries implicitly specify dependencies between stages and milestones: for instance, the sentry of the guard of stage Credit Check states that the stage is opened if milestone IDGS has been achieved, so the guard of Credit Check depends on IDGS. The dependencies are graphically depicted using dashed arrows in Fig. 1. (Our diagrammatic convention does not explicitly indicate how multiple milestones are combined in a sentry, e.g., the sentry for PCS; please refer to the tables.) Rectangles represent data attributes. A dashed line from a stage to a data attribute indicates that the stage computes a value for the data attribute. To compare different GSM schemas, we make use of output attributes, depicted in bold italics, which can be milestones or data attributes. Some attributes are not shown (spez., *fast_turnaround* computed by Fast Turnaround Business Eligibility and *BP_good* computed by Business Performance Evaluation Check).

The behavior of GSM schemas is driven by event occurrences, which are typically the result of completion of a stage execution. In response to an event occurrence, a B(usiness)-step is taken, in which as many sentries as possible are applied. For instance, suppose that in some “snapshot”, i.e., the state of an artifact instance at some time during its execution, the milestone BPECS is true and stages Credit Check and Fast Turnaround Bonus Eligibility are the

Table 2. Milestones for BCA^{base} in Fig. 1

Milestone	Full name	Sentry
IDGS	Initial Data Gathering Successful	C:Initial Data Gathering \wedge ...
IDGU	Initial Data Gathering Unsuccessful	C:Initial Data Gathering \wedge ...
CCS	Credit Check Successful	C:Credit Check \wedge rating \geq 8
CCU	Credit Check Unsuccessful	C:Credit Check \wedge rating $<$ 8
BPECS	Business Performance Evaluation Check Successful	C:Business Performance Evaluation Check \wedge BP_good
BPECU	Business Performance Evaluation Check Unsuccessful	C:Business Performance Evaluation Check \wedge \neg BP_good
PCS	Pre-checks Successful	CCS \wedge BPECS
PCU	Pre-checks Unsuccessful	CCU \vee BPECU
DCS	Detailed Check Successful	C:Detailed Check \wedge ...
DCU	Detailed Check Unsuccessful	C:Detailed Check \wedge ...
TBPS	Team Bonus Pay Successful	C:Team Bonus Pay

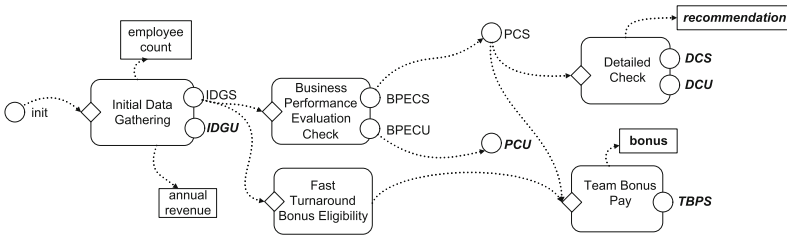


Fig. 2. Process BCA^{del} resulting after applying change of Example 2.3

only open stages. If stage completion event C:Credit Check now occurs with value 9 for rating, then milestone CCS gets achieved. The milestone PCS also gets achieved, and also stage Detailed Check is opened (and thus, the external activity associated with that stage is started). At this point no further sentries can be applied, the B-step is finished, and the new snapshot has been computed. (See also Example 3.5 below.) □

We next present three variations on this example.

Example 2.2. In another region, the business performance of a partner is evaluated and checked only if the partner has more than 300 employees. The GSM schema of Fig. 1 is changed as follows (the other sentries are not changed):

- The guard of stage Business Performance Evaluation Check becomes $IDGS \wedge \text{employee_count} \geq 300$.
- Milestone PCS can be achieved via extra sentry $CCS \wedge \text{employee_count} < 300$.

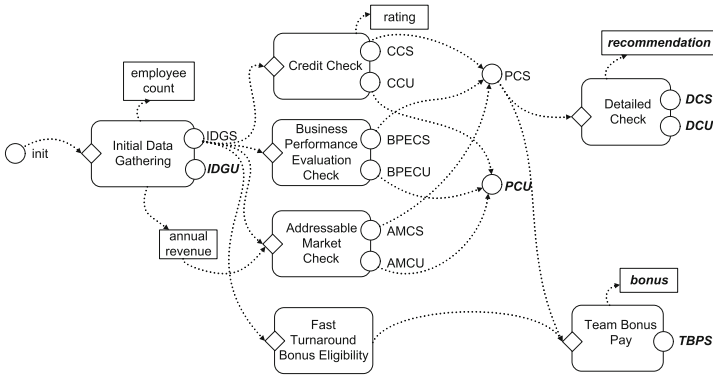


Fig. 3. Process BCA^{ins} after applying change of Example 2.4

Now the question arises how the change affects cases. We would like to assert that for partners with 300 or more employees, the new GSM schema emulates the behavior of the old GSM schema, and the old GSM schema emulates that of the new, so for the same cases, the same output results in both schemas. (‘Emulates’ is defined precisely in Definition 4.4 below.) For partners with less than 300 employees, this assertion does not hold. In particular, it may be that a company with say 290 employees and a poor performance is accepted under the new schema but rejected under the old schema. Example 5.2 will illustrate how the formalism and results of this paper can be applied to prove these properties. □

Example 2.3. Consider again the base process BCA^{base} of Example 2.1. In yet another region, the credit of the partner is not checked. Schema BCA^{base} is changed by deleting stage Credit check and milestones CCS and CCU, as visualized in BCA^{del} in Fig. 2. The sentries of milestones PCS and PCU need to change as follows (the other sentries are not changed):

- The sentry of milestone PCS becomes BPECS.
- The sentry of milestone PCU becomes BPECU.

To characterize the change, we would like to assert that for cases under the old schema for which the credit check was successful, the new schema emulates the old schema. For cases of partners for which the credit check was unsuccessful in Fig. 1 there is a difference: for those cases the detailed check can be performed as in Fig. 2. This example will be revisited in Examples 4.5 and 4.11. □

Example 2.4. Consider again the base process BCA^{base} . In a fourth region, the market addressed by the partner is assessed. Stage Addressable Market Check is inserted with milestones AMCS (Addressable Market Check Successful) and AMCU (Addressable Market Check Unsuccessful); see BCA^{ins} in Fig. 3. The sentries need to change as follows (the other sentries are not changed):

- The guard of stage Addressable Market Check becomes $\text{IGDS} \wedge \text{annual_revenue} \geq \$500K$;
- The sentry of milestone PCS is replaced with two sentries: $\text{CCS} \wedge \text{BPECS} \wedge \text{AMCS}$ and $\text{CCS} \wedge \text{BPECS} \wedge \text{annual_revenue} < \$500K$.
- The sentry of milestone PCU becomes $\text{CCU} \vee \text{BPECU} \vee \text{AMCU}$.

The change assertion is that for cases in which the annual revenue is lower than $\$500K$, the old schema emulates the new schema and vice versa. Also for cases in which the annual revenue is higher or equal to $\$500K$ and the milestone AMCS gets achieved, the old and the new schema emulate each other. This will be revisited in Example 5.9. \square

3 The Formal GSM Model

This section presents formal definitions for the variant of GSM used in this paper. This includes a specific notion of “executions” of a GSM schema, that will be important in our reasoning about property preservation. It is assumed that the reader is familiar with the basic aspects of the formal definitions of GSM (e.g., as in [6, 9, 14]).

The development here imposes a family of restrictions on the GSM variants of, e.g., [9, 14], to enable the development of interesting theoretical properties concerning schema evolution. A comparison of the GSM used here with previous variants is presented in [10]. Importantly, in the GSM variant used here, the executions are *monotonic*, that is, an attribute value does not change once it is defined. Generalization and adaptation of these results to richer variants of GSM, and to the full GSM model, are left for future research.

These assumptions enable a streamlined approach for the formal definitions of GSM schema and operational semantics.

We assume three infinite disjoint sets of names, for *data attributes*, for *milestones*, and for *stages*. Each data attribute a has a type $\text{type}(a)$ which is scalar (e.g., string, character, integer, float, etc.), or is a set of records of scalars. Milestones can be used as attributes with type Boolean. Both data attributes and milestones may take the unassigned (or null) value (denoted \perp).

We assume a condition language \mathcal{C} that includes fixed predicates over scalars (e.g., ‘ \leq ’ over integers or floats), and Boolean connectives. Quantification and testing set membership is supported for working with the set-valued attributes. The condition formulas may involve stage, milestone, and data attributes. All attributes start with undefined value (\perp). Milestones will take the value *True* if one of their sentries go true. Stages will take the value *True* at the time when they complete. (This is a variation on the traditional behavior of stage attributes.)

A *sentry* ψ has one of the three forms: “ φ ”, “ $\text{C} : S$ ”, or “ $\text{C} : S \wedge \varphi$ ”, where φ is a condition formula ranging over the attributes of Γ . Here “ $\text{C} : S$ ” is called the *completion event* for stage S . Also, $\text{C} : S$ (if present) is the *completion event* for ψ and φ (if present) is the *formula* for ψ .

Definition 3.1. A GSM schema is a 5-tuple $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{\text{start}}, Att_{\text{out}}, sen, sig)$ where:

1. Att_d is a finite set of data attributes.
2. Att_m is a finite set of milestone attributes.
3. Att_S is a finite set of stage attributes.
4. $m_{\text{start}} \in Att_m$ is called the *start milestone*. It is used as a mechanism for launching an execution of Γ .
5. $Att_{\text{out}} \subseteq Att_d \cup Att_m$ is the set of *output* attributes for Γ . This set is also denoted as $out(\Gamma)$.
6. The *sentry assignment* sen is a function from $Att_S \cup (att_m - \{m_{\text{start}}\})$ to sets of sentries with formulas in the condition language \mathcal{C} ranging over Att , and such that if there is a completion event $C : S$ then $S \in Att_S$.
7. The *signature assignment* sig is a function from Att_S to pairs (I, O) of finite sets of attributes from Att_d . If $sig(S) = (I, O)$, then we denote I as $sig_{\text{in}}(S)$, called the *input* of S , and denote O as $sig_{\text{out}}(S)$, called the *output* of S .

Sentries define dependencies between stage and milestone attributes of Γ . For $a_1, a_2 \in Att_S \cup Att_m$, a dependency (a_1, a_2) signifies that there is a sentry of a_2 that references a_1 . The dependency graph of Γ , denoted $DG(\Gamma)$, contains all these dependencies [10]. Schema Γ is *Fully Acyclic* if $DG(\Gamma)$ is a directed acyclic graph. Then Γ is called an *FA-GSM schema*. Each FA-GSM schema is a GSM schema in the sense of earlier work [6, 9], and the equivalence theorem for B-steps developed there also holds for FA-GSM schemas.

Definition 3.2. For a GSM schema $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{\text{start}}, Att_{\text{out}}, sen, sig)$ a *snapshot* is a mapping σ from Att into values of appropriate type (where some attributes may be assigned the null value \perp). For milestone and stage attributes, the only permitted values are \perp and *True*.

In the GSM model used here, milestone and stage attributes will never take the value *False*. This is because such attributes will remain undefined until they become true.

Definition 3.3. Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{\text{start}}, Att_{\text{out}}, sen, sig)$ be a GSM schema. Let μ be a sentry for milestone m , and let φ be the formula of μ . Given a snapshot σ of Γ (where some attributes may have undefined value), φ is *strictly satisfied* by σ , denoted $\sigma \models^{\text{strict}} \varphi$, if σ is non-null for each attribute A occurring in φ , and if φ is satisfied by σ .

Now let φ be the formula of a sentry for a stage S . For snapshot σ of Γ , φ is *strictly satisfied* for S by σ , denoted $\sigma \models_S^{\text{strict}} \varphi$, if (a) σ is non-null for each attribute in $sig_{\text{in}}(S)$, (b) σ is non-null for each attribute occurring in φ , and (c) φ is satisfied by σ .

In particular, if $\sigma \models_S^{\text{strict}} \varphi$, then each input attribute for S is defined, and so S can be launched. In this paper we focus on strict satisfaction, and refer to this simply as “satisfaction”.

The notion of *B-step* for a FA-GSM schema Γ and snapshot σ is defined as in [6, 9]. Further, it can be verified that the basic equivalence results from [6] apply to FA-GSM schemas.

Definition 3.4. Let Γ be an FA-GSM schema. An *execution* of Γ is a sequence

$$\xi = (\sigma_{init}, \sigma_0, \alpha_1, \beta_1, \sigma_1, \dots, \alpha_n, \beta_n, \sigma_n)$$

where (a) the σ 's are snapshots; (b) σ_{init} is special initial snapshot that is essentially all false except for m_{start} which is assumed to have just turned to *True*; (c) each σ_i is the result of a B-step based on the preceding σ and (for $i > 0$) the incoming stage completion, denoted as β_i . Stages may be launched as part of a B-step; the set of these are represented by the α 's. The family of executions of Γ is denoted $Exec(\Gamma)$.

An execution is *terminal* if it cannot be extended. The set of terminal executions is denoted $TermExec(\Gamma)$.

Example 3.5. We illustrate the notion of execution by revisiting Example 2.1 and the B-step described there. Snapshots are denoted here by listing all milestones that are true, all stages that are open, and the value of each defined data attribute. In each execution of BCA^{base} , $\sigma_0 = \{\text{Initial Data Gathering}\}$. After that stage completes, we might arrive at σ_1 that additionally has milestone IDGS true, and each of Credit Check, Business Performance Evaluation Check, and Fast Turnaround Bonus Eligibility open. Also, α_2 holds these three stage names. The next steps of the execution might be as follows.

$$\begin{aligned} \beta_2 &= \text{C :Business Performance Evaluation Check} \\ \sigma_2 &= \{\text{init, IDGS, BPECS,} \\ &\quad \text{Credit Check, Fast Turnaround Bonus Eligibility,} \\ &\quad \text{employee_count : 1200, annual_revenue : \$700K, BP_good : True}\} \\ \alpha_3 &= \emptyset \\ \beta_3 &= \text{C :Credit Check} \\ \sigma_3 &= \{\text{init, IDGS, BPECS, CCS, PCS,} \\ &\quad \text{Fast Turnaround Bonus Eligibility, Detailed Check,} \\ &\quad \text{employee_count : 1200, annual_revenue : \$700K,} \\ &\quad \text{rating : 9, BP_good : True,}\} \\ \alpha_4 &= \{\text{Detailed Check}\} \end{aligned}$$

The B-step of Example 2.1 occurs from β_3 to σ_3 . □

4 Reasoning About GSM Executions

This section develops tools for reasoning about GSM executions, including comparing the executions supported by different FA-GSM schemas. The first subsection introduces the notion of *stage i/o assignments*, used to formally study the possible behaviors of stage executions. The second subsection defines *conditional emulation*, which provides the basis for formally comparing the behaviors of FA-GSM schemas. And the third subsection presents the *Lifting Lemma*.

4.1 Stage i/o Assignments

A primary goal of this paper is to study the preservation of properties when transforming an FA-GSM schema Γ^1 into a related FA-GSM schema Γ^2 . To accomplish this we study properties of elements of $Exec(\Gamma^1)$ vis-a-vis elements of $Exec(\Gamma^2)$. *Non-determinism* in executions of an FA-GSM Γ may lead to different outcomes for the same input, which complicates a fair comparison among executions of different schemas. There are two ways that non-determinism arises:

Different Stage Outputs: Since many stages correspond to human activities, the outputs may vary due to a variety of factors that are not explicitly available in the snapshot that launched the stage containing that stage.

Different Stage Completion Timing: Because sentries may include stage completion events, there may be “race” conditions under which a sentry does or does not fire. For example, consider sentry $\psi = C : S \wedge \varphi$. Suppose that in a particular execution ξ stage S completes before all variables in φ have become defined. Then ψ can never be triggered in ξ . In contrast, if S completes after all variables in φ have become defined then ψ might trigger in ξ .

The next definition allows us to focus on pairs of executions for which all shared stages have the same behavior.

Definition 4.1. Given FA-GSM schema Γ a *stage i/o assignment* is a function τ with domain the stages of Γ , such that for each stage S , $\tau[S]$ is a function whose signature matches the signature of S in Γ .

An execution $\xi = \sigma_{init}, \sigma_0, \alpha_1, \beta_1, \dots, \sigma_n$ of Γ is *compliant* with τ if for each $i \in [1..n]$, the payload of the stage completion $\beta_i = C : S(c_1, \dots, c_p)$ corresponds to the application of $\tau[S]$ on the values from the snapshot that launched S .

Example 4.2. Let IDG denote stage Initial Data Gathering of BCA^{base} , and BPEC denote Business Performance Evaluation Check. In one stage i/o assignment τ_{ABC} for the *ABC* company, we might have

$$\begin{aligned}\tau_{ABC}[\text{IDG}](\text{employee_count}) &= 1200 \\ \tau_{ABC}[\text{IDG}](\text{annual_revenue}) &= \$500K \\ \tau_{ABC}[\text{BPEC}](\text{BP_good}) &= \textit{True}\end{aligned}$$

Because the evaluations of business performance may be subjective, a different stage i/o assignment τ'_{ABC} might arise, with $\tau'_{ABC}[\text{IDG}] = \tau_{ABC}[\text{IDG}]$ but $\tau'_{ABC}[\text{BPEC}](\text{BP_good}) = \textit{False}$. \square

We use the stage i/o assignment to “explain” the output of stages in a particular execution. Intuitively, the following result states that the full range of non-determinism in GSM executions can be controlled by holding the stage behaviors and the relative timing of stage completion fixed (proof omitted).

Lemma 4.3. If two executions of Γ are compliant with the same stage i/o assignment, and if the order of stage completions is the same, then they are identical in all other ways as well.

4.2 Conditional Emulation

In the general case, we shall be looking at a pair Γ^1, Γ^2 of FA-GSM schemas and attempting to compare elements of $TermExec(\Gamma^1)$ with elements of $TermExec(\Gamma^2)$. We typically focus on executions that satisfy a condition, e.g., in the case of Example 2.2, $\Omega = \text{“employee count} \geq 300\text{”}$. We then demonstrate that executions of one schema that satisfy the condition can be emulated by executions of the other, e.g., for each execution of BCA^{mod} that satisfies Ω there is a corresponding execution of BCA^{base} that behaves identically on output attributes PCU, DCS, and recommendation (see Example 5.2 below).

In the sequel, if f is a function over domain D , and $C \subseteq D$, then $f|_C$ denotes the restriction of f to C .

Suppose now that $\Gamma^i = (Att^i = Att_d^i \cup Att_m^i \cup Att_s^i, m_{start}^i, Att_{out}^i, sen^i, sig^i)$ for i in $[1,2]$. Suppose further that τ^i is a stage i/o assignment for Γ^i , i in $[1,2]$. Then τ^1 and τ^2 are *compatible* if $\tau^1|_{Att_5^1 \cap Att_5^2} = \tau^2|_{Att_5^1 \cap Att_5^2}$.

Let Γ^1, Γ^2 be as above. As suggested above, we shall work with conditions Ω over the union $Att^1 \cup Att^2$, in order to focus on executions of Γ^1 or Γ^2 of interest. For a snapshot σ^1 over Γ^1 , σ^1 *satisfies* Ω with existential extension, denoted $\sigma^1 \models^{ex} \Omega$, if there is some extension σ of σ^1 to include all attributes of Ω not in Att^1 , such that $\sigma \models^{strict} \Omega$.

We now define the notion of “conditional emulatability”, which enables us to compare the behavior of pairs of schemas with regards to selected attributes.

Definition 4.4. Let $\Gamma^i = (Att^i = Att_d^i \cup Att_m^i \cup Att_s^i, m_{start}^i, Att_{out}^i, sen^i, sig^i)$ be an FA-GSM schema for i in $[1,2]$, and let $\mathcal{A} \subseteq Att^1 \cap Att^2$, and let Ω be a condition over $Att^1 \cup Att^2$. Then Γ^1 *emulates* Γ^2 under Ω , denoted $\Gamma^1 \rightarrow_{\Omega, \mathcal{A}} \Gamma^2$, if the following holds. If

1. τ^2 is a stage i/o assignment for Γ^2 ;
2. $\xi^2 \in Exec(\Gamma^2)$ is a (possibly non-terminal) τ^2 -compliant execution with final snapshot σ^2 ; and
3. $\sigma^2 \models^{ex} \Omega$

then

1. there exists a stage i/o assignment τ^1 for Γ^1 that is compatible with τ^2 , and
2. there exists a τ^1 -compliant execution $\xi^1 \in Exec(\Gamma^1)$ with final snapshot σ^1 ,
3. such that $\sigma^1|_{\mathcal{A}} = \sigma^2|_{\mathcal{A}}$.

We write $\Gamma^1 \rightleftharpoons_{\Omega, \mathcal{A}} \Gamma^2$ if $\Gamma^1 \rightarrow_{\Omega, \mathcal{A}} \Gamma^2$ and $\Gamma^2 \rightarrow_{\Omega, \mathcal{A}} \Gamma^1$.

Example 4.5. Recall BCA^{base} (Example 2.1) and BCA^{del} (Example 2.3). Let $\mathcal{A} = \text{PCS, PCU}$ and $\Omega = \text{“Rating} = 9\text{”}$. We illustrate now how it can be shown that $\Gamma^1 \rightleftharpoons_{\Omega, \mathcal{A}} \Gamma^2$. For the \rightarrow direction, fix stage i/o assignment τ^2 for Γ^2 . We focus here on executions ξ^2 of Γ^2 where IDGS is satisfied. In those cases, the only τ^1 that extends τ^2 and enables satisfaction of Ω will have $\tau^1[\text{Credit Check}](\text{Rating}) = 9$. For this τ^1 , the stage Credit Check will execute and return Rating with value 9 and trigger the milestone CCS. Thus, an execution ξ^1 compliant with

τ^1 can be constructed from ξ^2 by inserting the launch and completion of Credit Rating sometime in between the satisfaction of IDGS and satisfaction of CCS. Emulation in the other direction is straightforward to show. \square

4.3 The Lifting Lemma

The Lifting Lemma will enable us to infer emulatability in terms of output attributes, i.e., at a “global level”, based on emulatability in terms of selected milestone attributes, i.e., at a “local level”.

To state the lifting lemma we need to be able to talk about the areas where schemas Γ^1, Γ^2 differ.

Definition 4.6. Let Γ^1, Γ^2 be FA-GSM schemas, and let Δ^i be a subset of the stages and milestones of Γ^1 for i in $[1,2]$. Then Δ^1, Δ^2 is a *change pair* for Γ^1, Γ^2 if the two schemas are identical except for the milestones and stages (and their sentries) in the delta’s.

Next, we introduce the notion of “fence” that allows us to create a separation between a change set and an output attribute.

Definition 4.7. Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_s, m_{start}, Att_{out}, sen, sig)$ be an FA-GSM schema, let $\Delta \subset Att_m \cup Att_s$, and let $\mathcal{O} \subseteq Att_{out}$. A set $\mathcal{F} \subseteq Att_m$ is a *fence* between Δ and \mathcal{O} if for each pair $\delta \in \Delta, o \in \mathcal{O}$ and each path ρ from δ to o in $DG(\Gamma)$ there is some $m \in \mathcal{F}$ on path ρ .

Speaking intuitively, if \mathcal{F} is a fence between Δ and \mathcal{O} , and if certain “race” conditions do not hold, then the values assigned to \mathcal{O} will not be impacted by the behavior in the Δ area. The next definition identifies the “race” conditions that need to be avoided (see Example 4.9 below).

Definition 4.8. Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_s, m_{start}, Att_{out}, sen, sig)$ be an FA-GSM schema, $\mathcal{F} \subseteq Att_m$ a set of milestones in Γ , and $v \in Att_m \cup Att_s$. Then v is *completion independent* modulo \mathcal{F} if for each stage $S \in Att_s$ and each path ρ from S to v , if there is a node w on ρ with a sentry of form “C :S . . .”, then there is a node $f \in \mathcal{F}$ that lies between w and v in ρ .

Example 4.9. In BCA^{base} , with the exception of bonus and TBPS, all output attributes are completion independent modulo $\{PCS\}$. In contrast, bonus and TBPS are not, because of the completion event C:Fast Turnaround Bonus Eligibility in the guard for stage Team Bonus Pay. \square

We now have the Lifting Lemma, which states that under certain conditions, if Γ^1 emulates Γ^2 for the elements of a fence, then Γ^1 also emulates Γ^2 for output attributes that are downstream from that fence. The proof, omitted, is based on splicing of executions.

Lemma 4.10 (Lifting Lemma). Let $\Gamma^i = (Att^i = Att_d^i \cup Att_m^i \cup Att_S^i, m_{start}^i, Att_{out}^i, sen^i, sig^i)$ be an FA-GSM schema for i in $[1,2]$. Suppose that:

1. Δ^1, Δ^2 is a change pair for Γ^1, Γ^2 .
2. $\mathcal{O} \subseteq out(\Gamma^1) \cap out(\Gamma^2)$.
3. \mathcal{F} is a fence between Δ^i and \mathcal{O} in Γ^i for i in $[1,2]$.
4. \mathcal{O} is completion independent modulo \mathcal{F} in Γ^i , for i in $[1,2]$.
5. Ω is a condition over $Att^1 \cup Att^2$.
6. $\Gamma^1 \rightarrow_{\Omega, \mathcal{F}} \Gamma^2$.

Then $\Gamma^1 \rightarrow_{\Omega, \mathcal{O}} \Gamma^2$.

We next apply the Lifting Lemma to the example of deletion from Sect. 2.

Example 4.11. Recall Example 4.5, and the property $BCA^{base} \rightleftharpoons_{\Omega, \mathcal{A}} BCA^{del}$, where $\mathcal{F} = \{\text{PCS, PCU}\}$ and $\Omega = \text{“rating = 9”}$. Let $\Delta^1 = \{\text{Credit Check, CCS, CCU, PCS, PCU}\}$ and $\Delta^2 = \{\text{PCS, PCU}\}$. Then Δ^1, Δ^2 is a change pair for Γ^1, Γ^2 . It is straightforward to verify that \mathcal{F} is a fence between these change sets and the output attributes $\mathcal{O} = \{\text{IDGU, PCU, recommendation, DCS, DCU}\}$. Thus, by the Lifting Lemma, $\Gamma^1 \rightleftharpoons_{\Omega, \mathcal{O}} \Gamma^2$. Intuitively, this states that Γ^1, Γ^2 have identical behavior on \mathcal{O} , if the rating attribute is assumed to have value 9. There are no guarantees with regards to the attribute bonus), because of a possible race condition involving the completion of Fast Turnaround Bonus Eligibility, which occurs in the sentry for Team Bonus Pay.

However, note that bonus and TBPS have a completion dependency on Fast Turnaround Bonus Eligibility that is not blocked by \mathcal{F} . As a result, the Lifting Lemma does not apply to those attributes. Indeed, it is possible to construct an example execution ξ^1 of Γ^1 where Team Bonus Pay is not launched, but in the corresponding execution ξ^2 of Γ^2 this stage would launch.

Note that if the completion event C:Fast Turnaround Bonus Eligibility in the guard for Team Bonus Pay were dropped, then Term Bonus Pay, TBPS, and bonus would be completion independent modulo \mathcal{F} , and so the Lifting Lemma would apply to them. \square

5 Property Preserving Schema Modifications

This section presents operators for modifying FA-GSM schemas that guarantee the preservation of various properties. The operators focus on sentry modification, and on deletions and insertions of stages and milestones. The proofs about property preservation rely on the Lifting Lemma. (The proofs are omitted here, but available in [10].) Examples from Sect. 2 are used to illustrate the results developed here.

We begin with a useful observation that is a very straightforward consequence of the Lifting Lemma. Before making the observation, we need the following: the notion of “shadow” of a change set. Specifically, the *shadow* of Δ is the set of milestones, stages, and data attributes that are “downstream” of nodes in Δ in the graph $DG(\Gamma)$.

Let Δ^1, Δ^2 be a change pair for FA-GSM schemas Γ^1, Γ^2 . It is easily shown that $shadow(\Delta^1, \Gamma^1) = shadow(\Delta^2, \Gamma^2)$.

Proposition 5.1. *Let $\Gamma^i = (Att^i = Att_d^i \cup Att_m^i \cup Att_S^i, m_{start}^i, Att_{out}^i, sen^i, sig^i)$ for i in $[1,2]$, and let Δ^1, Δ^2 be a change pair for Γ^1, Γ^2 . Let $\mathcal{A} = shadow(\Delta^1, \Gamma^1) = shadow(\Delta^2, \Gamma^2)$, and let $\mathcal{O} = (Att_{out}^1 \cup Att_{out}^2) - \mathcal{A}$. Then $\Gamma^1 \rightleftharpoons_{True, \mathcal{O}} \Gamma^2$.*

We next examine a simple form of sentry modification.

Example 5.2. Consider BCA^{base} from Example 2.1 and BCA^{mod} from Example 2.2. Recall that BCA^{mod} is formed from BCA^{base} by modifying the sentry on Business Performance Evaluation Check, to skip launching of that stage if the client has < 300 employees, and adding a sentry for milestone PCS. Let $\Omega = \text{“employee_count} \geq 300\text{”}$. A case-by-case argument can be used to show that $BCA^{base} \rightleftharpoons_{\Omega, \{PCS, PCU\}} BCA^{mod}$. Now let

- $\mathcal{O} = \{IDGU, PCU, recommendation, DCS, DCU\}$.
- $\Delta^1 = \{\text{Business Performance Evaluation Check, BPECS, BPECU, PCS}\}$.
- $\Delta^2 = \{PCS\}$.

Similar to Example 4.11, it is easily verified that $\mathcal{F} = \{PCS, PCU\}$ is a fence for Δ^i and \mathcal{O} , for i in $[1,2]$. Further, \mathcal{O} is completion-independent modulo \mathcal{F} . The Lifting Lemma now implies that $\Gamma^1 \rightleftharpoons_{\Omega, \mathcal{O}} \Gamma^2$. \square

5.1 Deletion

This subsection develops constructions for deleting milestones and stages from FA-GSM schemas. Similar to the examples of Sect. 2, the focus is on enabling the deletions while maximizing emulatability.

We begin by describing the construction for deleting a single milestone. We shall use two notational conventions. The first is for substitutions in sentries: given a sentry ψ , an attribute z , and a formula φ , $\psi[z/\varphi]$ denotes the result of replacing all occurrences of z in ψ by (φ) . The second is a manipulation on sentries called *completion-event removal*: For a sentry of form $\psi = C : S \wedge \varphi$, define $cer(\psi)$ to be $S \wedge \varphi$. Notice that ψ will be true for the single B-step where stage S completes, whereas $cer(\psi)$ will be true for that B-step and all subsequent B-steps. If ψ does not include a completion event, If ψ is eventless, then $cer(\psi) = \psi$.

The following definition specifies implicitly an algorithm for deleting a milestone while preserving all output behaviors.

Definition 5.3. Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{start}, Att_{out}, sen, sig)$ be an FA-GSM schema, m a milestone of Γ , and $M = \{\psi_1, \dots, \psi_q\}$ the set of sentries of m in Γ . The *deletion* of m from Γ , denoted $del(\Gamma, m)$, is the FA-GSM schema constructed from Γ in the following way. Suppose that v is a stage or milestone in Γ , that χ is a sentry for v , and that m occurs in χ . Then replace χ in Γ with a set of sentries

$$N = \{\chi[m/cer(\psi_p)] \mid p \in [1, q]\}$$

Finally, delete m from the set of attributes of Γ .

Intuitively, in the construction of schema $del(\Gamma, m)$ occurrences of m in sentries are replaced by “macro-expansions” of m . It can be shown that in this construction, the set \mathcal{F} of stages and milestones whose sentries are changed can serve as a fence, and that $\Gamma \rightleftharpoons_{True, \mathcal{F}} del(\Gamma, m)$.

Deleting a stage S from an FA-GSM schema Γ is similar to deleting a milestone, in terms of performing “macro-expansions” in selected sentries. However, there are three complications. First, data attributes produced by S are assigned default values \vec{c} . Second, use of the default values must be delayed until S would have completed. And third, sentries that include S or $C :S$ must be rewritten.

Definition 5.4. Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{start}, Att_{out}, sen, sig)$ be an FA-GSM schema, S a stage of Γ , and $M = \{\psi_1, \dots, \psi_q\}$ the set of sentries of m in Γ . Let $\vec{a} = sig_{out}(S)$ and let \vec{c} be a vector of constants having types that match \vec{a} . The *deletion* of S from Γ using \vec{c} for \vec{a} , denoted $del(\Gamma, S, \vec{a}/\vec{c})$, is an FA-GSM schema constructed from Γ in the following way. Suppose that v is a stage or milestone in Γ , that χ is a sentry for v , and that χ includes $C :S$ and/or includes one or more attribute from \vec{a} . Then replace χ with a set of sentries

$$N = \{\chi[C :S/\psi_p, \vec{a}/\vec{c}] \wedge cer(\psi_p) \mid p \in [1, q]\}.$$

Finally, delete S from Att_S .

Example 5.5. To illustrate the above construction, consider a variation BCA_{var}^{del} of BCA^{del} , in which only the stage Credit Check is deleted, but milestones CCS and CCU are to be retained. In this case, the sentry of CCS will become “IDGS $\wedge 9 \geq 8$ ”, and the sentry of CCU will become “IDGS $\wedge 9 < 8$ ”. \square

Suppose that \mathcal{F} is the set of milestones and stages whose sentries are impacted by the stage deletion, if \vec{c} is a vector of constants having the types of $\vec{a} = sig_{out}(S)$, and let Ω be “ $\vec{a} = \vec{c}$ ”. Then $\Gamma \rightleftharpoons_{\Omega, \mathcal{F}} del(\Gamma, m)$.

We now state a general result concerning deletion of a set \mathcal{X} of stages and milestones. Let \vec{a}/\vec{c} be the union of all the mappings from attributes to constants (used for the stages that are deleted). The delete operators above can be applied one at a time for the elements of \mathcal{X} , and the ordering does not affect the end result, denoted as $del(\Gamma, \mathcal{X}, \vec{a}/\vec{c})$.

Theorem 5.6. Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{start}, Att_{out}, sen, sig)$, \mathcal{X} , \vec{a} and \vec{c} and $\Gamma' = del(\Gamma, \mathcal{X}, \vec{a}/\vec{c})$ be as above. Let \mathcal{F} be the collection of all stages and milestones in Γ' whose sentries have been modified, and let Ω be the formula $\vec{a} = \vec{c}$. Then $\Gamma \rightleftharpoons_{\Omega, \mathcal{F}} \Gamma'$. Furthermore, if $\mathcal{O} \subseteq Att_{out}$ is completion-independent modulo \mathcal{F} , then $\Gamma \rightleftharpoons_{\Omega, \mathcal{O}} \Gamma'$.

5.2 Insertion

This subsection studies property preservation in the context of insertions to an FA-GSM schema Γ . Speaking intuitively, the emphasis here is on enabling the designer to insert one or several stages and milestones, while ensuring that the

global impact of the insertion is minimized “when things go right” (cf. schema BCA^{ins}).

The following definition is provided to talk about “bulk” insertions.

Definition 5.7. Let Γ be an FA-GSM schema. An *insertable fragment* for Γ is a tuple $\Delta = (Att^\Delta = Att_d^\Delta \cup Att_m^\Delta \cup Att_S^\Delta, Att_{out}^\Delta, sen^\Delta, sig^\Delta)$ where the set Att^Δ are new, where the sentries of Δ may refer to attributes from Γ and Δ , and where the insertion of Δ into Γ yields a well-formed FA-GSM schema.

To enable modular insertions, and to facilitate straightforward reasoning about the impact of an insertion, a best practice is to include as part of Δ one or more milestones that are used to indicate the “success” or “failure” of a case with regards to the inserted activity. The following result assumes there is a single “success” milestone. The result follows easily from the Lifting Lemma.

Theorem 5.8. Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{start}, Att_{out}, sen, sig)$ be an FA-GSM schema, and let $\Delta = (Att^\Delta = Att_d^\Delta \cup Att_m^\Delta \cup Att_S^\Delta, Att_{out}^\Delta, sen^\Delta, sig^\Delta)$ be an insertable fragment that includes a milestone $m_{success}$. Suppose that $\mathcal{F} \subseteq Att_m$ is a family of milestones in Γ , and suppose that Γ' is the result of modifying $ins(\Gamma, \Delta)$ by replacing each sentry μ of a milestone in \mathcal{F} by $\mu \wedge m_{success}$. Let $\Omega = “m_{success}”$. Finally, let $\mathcal{O} \subseteq Att_{out}$ be completion-independent modulo \mathcal{F} in Γ' . Then $\Gamma \xrightarrow{\Omega, \mathcal{O}} \Gamma'$.

Example 5.9. In the schema BCA^{ins} of Example 2.4, with regards to the above theorem, the milestone AMCS plays the role of $m_{success}$, the set {PCS} plays the role of \mathcal{F} , and the set {recommendation, DCS, DCU} plays the role of \mathcal{O} . In this case, the theorem tells us that for each execution of BCA^{ins} for which AMCS goes true, there is a corresponding execution of the base schema BCA^{base} with the same outcomes on \mathcal{O} . \square

6 Related Work

We discuss the literature on changes in process models for activity-centric business process management and case management.

In the context of *activity-centric BPM*, change operations have been proposed [26]. Different correctness criteria have been identified in the literature to assess which changes are allowed so that cases can be migrated properly from an old to a new schema [24]. A particular focus has been on ensuring that when the execution of a BP instance starts on one schema and migrates to another one while in flight, the final BP instance corresponds to an execution of the new schema. In our approach, we study a novel form of correctness, which focuses on preservation of schema properties, defined in terms of emulatability of one schema by another one. A form of unconditional emulatability was studied in connection with declarative artifact-centric business processes in [4]. That work was in an abstract setting; in contrast the results here are tied to a practical Case Management model, and motivated by a real-world use case.

Case management originates from industry, including, e.g., [25] and work on business artifacts, e.g., [22]. Recent overview works include [8, 13, 20]. Case management is related to the more general concept of data-centric business process management, which studies how activity-centric processes can be made more data-aware [16, 18, 22, 23] to improve their flexibility. This includes work on declarative artifact-centric models, including GSM [5, 6] and declarative process models for case management [12].

Though the problem of change has been recognized as central to case management [13], in particular adaptive case management [19], it has not been widely studied. Mukkalama et al. [21] study change in DCR Graphs, a declarative formalism for case management. They define basic change operations that add and remove behavior, but their operations are aimed at a micro-level, so removing atomic elements from schema. In our approach, we study also the impact of adding and removing larger fragments, so at a macro level. They focus on logical correctness and the use of automated verification techniques, whereas we develop tests for property preservation that can be checked at a syntactic level.

Motahari et al. [19] present a framework and prototype implementation that supports adaptive case management in social enterprises. The framework supports change, but does not address preservation of properties across changes.

There has been active research on verification for artifact-centric BPM models (e.g., [1, 2, 7, 11]). That work could be also be applied to reason about preservation of properties of case management schemas during evolution. The approach in the current paper uses syntactic conditions rather than semantic ones, and would thus be substantially easier to deploy and maintain than a verification-based approach.

7 Conclusion

This paper studies schema modifications in the context of a variant of the Guard-Stage-Milestone (GSM) model for Case Management. The main contributions of this paper are (i) a precise definition for testing the preservation of properties through the use of conditional emulatability; (ii) the development of a general-purpose “Lifting Lemma” which allows a variety of approaches to achieve and/or prove property preservation; and (iii) the specification of operators to perform schema manipulations that are guaranteed to preserve certain properties. The theoretical work is motivated by examples arising in a real-world application.

The research here can be extended in several directions, including the following: (a) extend results to more general kinds of GSM schema; (b) extend results to other Case Management and BPM models ([25] is a natural first candidate, and also the OMG CMMN standard [3]); (c) develop algorithms for schema modifications other than deletion and insertion, that preserve specified properties; (d) generalize to support adaptation of schemas for cases that are “in-flight”. and (e) develop approaches to apply the theoretical results developed here in practical settings.

References

1. Belardinelli, F., Lomuscio, A., Patrizi, F.: Verification of GSM-based artifact-centric systems through finite abstraction. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) *Service Oriented Computing*. LNCS, vol. 7636, pp. 17–31. Springer, Heidelberg (2012)
2. Bhattacharya, K., Gerede, C.E., Hull, R., Liu, R., Su, J.: Towards formal analysis of artifact-centric business process models. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) *BPM 2007*. LNCS, vol. 4714, pp. 288–304. Springer, Heidelberg (2007)
3. BizAgi and others. *Case Management Model and Notation (CMMN)*, v1, May 2014. OMG Document Number formal/2014-05-05, Object Management Group
4. Calvanese, D., De Giacomo, G., Hull, R., Su, J.: Artifact-centric workflow dominance. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) *ICSOC-ServiceWave 2009*. LNCS, vol. 5900, pp. 130–143. Springer, Heidelberg (2009)
5. Cohn, D., Hull, R.: Business artifacts: a data-centric approach to modeling business operations and processes. *IEEE Data Eng. Bull.* **32**(3), 3–9 (2009)
6. Damaggio, E., Hull, R., Vaculín, R.: On the equivalence of incremental and fixpoint semantics for business artifacts with guard-stage-milestone lifecycles. *Inf. Syst.* **38**, 561–584 (2013)
7. Deutsch, A., Hull, R., Patrizi, F., Vianu, V.: Automatic verification of datacentric business processes. In: *Proceedings of the International Conference on Database Theory (ICDT)* (2009)
8. Di Ciccio, C., Marrella, A., Russo, A.: Knowledge-intensive processes: characteristics, requirements and analysis of contemporary approaches. *J. Data Semant.* **4**(1), 29–57 (2015)
9. Eshuis, R., Hull, R., Sun, Y., Vaculín, R.: Splitting GSM schemas: a framework for outsourcing of declarative artifact systems. *Inf. Syst.* **46**, 157–187 (2014)
10. Eshuis, R., Hull, R., Yi, M.: Reasoning about Property Preservation in Adaptive Case Management. BETA Working Paper Series, Eindhoven University of Technology (2015)
11. Hariri, B.B., Calvanese, D., Giacomo, G.D., Deutsch, A., Montali, M.: Verification of relational data-centric dynamic systems with external services. In: *Proceedings of the International Symposium Principles of Database Systems*, pp. 163–174 (2013)
12. Hildebrandt, T.T., Mukkamala, R.R., Slaats, T.: Designing a cross-organizational case management system using dynamic condition response graphs. In: *Proceedings of the IEEE EDOC 2011*, pp. 161–170. IEEE Computer Society (2011)
13. Huber, S., Hauptmann, A., Lederer, M., Kurz, M.: Managing complexity in adaptive case management. *Proc. S-BPM ONE* **360**, 209–226 (2013)
14. Hull, R., Damaggio, E., Masellis, R.D., Fournier, F., Gupta, M., III, F.H., Hobson, S., Linehan, M., Maradugu, S., Nigam, A., Sukaviriya, P., Vaculín, R.: Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events. In: *Proceedings of the 5th ACM International Conference on Distributed Event-Based Systems*, pp. 51–62. DEBS, USA (2011)
15. Hull, R., Narendra, N.C., Nigam, A.: Facilitating workflow interoperability using artifact-centric hubs. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) *ICSOC-ServiceWave 2009*. LNCS, vol. 5900, pp. 1–18. Springer, Heidelberg (2009)
16. Künzle, V., Reichert, M.: Philharmonicflows: towards a framework for object-aware process management. *J. Softw. Maintenance* **23**(4), 205–244 (2011)

17. Limonad, L., Boaz, D., Hull, R., Vaculín, R., Heath, F.T.: A generic business artifacts based authorization framework for cross-enterprise collaboration. In: SRII Global Conference, pp. 70–79 (2012)
18. Meyer, A., Pufahl, L., Fahland, D., Weske, M.: Modeling and enacting complex data dependencies in business processes. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 171–186. Springer, Heidelberg (2013)
19. Motahari-Nezhad, H.R., Bartolini, C., Graupner, S., Spence, S.: Adaptive case management in the social enterprise. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) Service Oriented Computing. LNCS, vol. 7636, pp. 550–557. Springer, Heidelberg (2012)
20. Motahari-Nezhad, H.R., Swenson, K.D.: Adaptive case management: overview and research challenges. In: IEEE Conference on Business Informatics (CBI) 2013, pp. 264–269. IEEE (2013)
21. Mukkamala, R.R., Hildebrandt, T.T., Slaats, T.: Towards trustworthy adaptive case management with dynamic condition response graphs. Proc. EDOC **2013**, 127–136 (2013)
22. Nigam, A., Caswell, N.S.: Business artifacts: an approach to operational specification. IBM Syst. J. **42**(3), 428–445 (2003)
23. Redding, G., Dumas, M., ter Hofstede, A.H.M., Iordachescu, A.: A flexible, object-centric approach for business process modelling. SOCA **4**(3), 191–201 (2010)
24. Rinderle, S., Reichert, M., Dadam, P.: Correctness criteria for dynamic changes in workflow systems - a survey. Data Knowl. Eng. **50**(1), 9–34 (2004)
25. van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. Data Knowl. Eng. **53**(2), 129–162 (2005)
26. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features - enhancing flexibility in process-aware information systems. Data Knowl. Eng. **66**(3), 438–466 (2008)

Cloud Services (Short Papers)

Modelling and Optimizing Bandwidth Provision for Interacting Cloud Services

Chao Chen¹, Ligang He^{1,2(✉)}, Bo Gao¹, Cheng Chang², Kenli Li²,
and Keqin Li^{2,3}

¹ Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK
{[@dcs.warwick.ac.uk](mailto:chao,liganghe,bogao)}

² School of Computer Science and Electronic Engineering, Hunan University,
Changsha 410082, China
{[@hnu.edu.cn](mailto:chengchang,kl1)}

³ Department of Computer Science, State University of New York,
New Paltz, NY 12561, USA
lik@newpaltz.edu

Abstract. Non-deterministic communication patterns among interacting Cloud services impose a challenge in determining appropriate bandwidth provision to satisfy the communication demands. This paper aims to address this challenge and develops a Communication Input-Output (CIO) model to capture data communication produced by Cloud services. The proposed model borrows the ideas from the Leontief's Input-Output Model in economy. Based on the model, this paper develops a method to determine the bandwidth provision for individual VMs that host a service. We further develop a Communication-oriented Simulated Annealing (CSA) algorithm, which takes an initial VM-to-PM mapping as input and finds the mapping with the minimal bandwidth provision and without increasing the PM usage in the initial mapping. Experiments have been conducted to evaluate the effectiveness and efficiency of the CIO model and the CSA algorithm.

1 Introduction

Services deployed in a Cloud are often hosted in a number of virtual machines (VMs), which are then placed on Physical Machines (PMs). In a Cloud environment, when services are invoked, the service invocations are often not isolated. An invocation to a service may spawn further invocations to other services. Moreover, service invocation and consequently data communication among them may not be static, but depend on the dynamic system information or service input. Consider the following example. NASDAQ QMX, the largest stock exchange company in the world, has been developing their data analysis services on Amazon Web Services (AWS) [1]. The data analysis process may involve a collection of interacting services, which are implemented through the standard services

This research was partially funded by the Key Program of National Natural Science Foundation of China (Grant Numbers: 61133005, 61432005).

provided in AWS, such as S3, VPC, EC2, etc. Which services are involved in the data analysis workflow and their invocation order are not static, but depend on dynamic system information at runtime, such as the initial data submitted by the clients, performance or security needs of the clients, and so on.

This brings the challenge to determine the bandwidth provision for these services and more specifically for the VMs that host the services. Solving the problem of VM bandwidth provision can help the tenants equip the VMs with proper communication capacity. In EC2, different types of VM instances have different communication capacity and consequently different price rates. Moreover, the data transfer between VMs is also charged in AWS. An exemplar application of this work is that when an enterprise tenant purchases the VMs in EC2 to build a business Cloud platform, offering to its users a rich set of interacting services, this work can help the enterprise decide which type of VM instance is most appropriate for each service, so that the VMs are able to fulfil the communication requirement inherent in the business Cloud while the enterprise does not pay unnecessary extra bills for VMs with higher bandwidth.

This paper aims to address this challenge by developing a Communication Input-Output (CIO) model for data communication among services. It borrows the idea from Leontief's Input-Output model in Economy and captures the interaction relation and impact among services. The data communication performed by each service can be calculated from the model. Knowing data communication performed by a service does not necessarily mean that the solution is apparent to the problem of bandwidth provision for the service's VMs. This is because if two VMs of two communicating services are consolidated into the same PM, the data transmission between these two VMs does not consume their bandwidth. Generally, even if the bandwidth provision for the services is determined, the bandwidth provision for each individual VM still depends on the specific VM-to-PM mapping. A lot of existing work has investigated the methods to find the VM-to-PM mapping with the minimal number of PMs. However, previous work does not take into account the non-deterministic nature of service interaction when they design their consolidation strategies. Our studies found that even if the VM-to-PM mapping has the minimal number of PMs, there is still room to further reduce the communication cost in the mapping while maintaining the minimal number of PMs. This paper designs and implements a Communication-oriented Simulated-Annealing (CSA) algorithm to reduce the total bandwidth provision of all VMs in a set of interacting services. The CSA algorithm takes as input the VM-to-PM mapping with the minimal number of PMs that is generated by the existing strategies. The CSA gradually adjusts the initial VM mapping to generate new mappings with reduced bandwidth provision. The adjustment of VM mappings is designed in the way that it does not increase the number of used PMs.

2 Background and Related Work

Background of the IO Model. Leontief's input-output (IO) model [2] divides an economy into sectors (e.g. agriculture, manufacturing, etc.). Goods produced

by a sector are consumed by the consumer market and other sectors. The consumer market is referred to as the *open sector*. Demands generated from the open sector are referred to as *external demands*. Goods that are exchanged between sectors is referred to as *internal demands* of the economy. Let column vectors A and X denote the external demands and the total demands of all sectors in the economy respectively, and C denote the internal consumption matrix of the economy in which c_{ij} represents the amount of goods that need to be consumed by sector i to produce one unit of goods in sector j . Leontief's IO model can be expressed by Eq. 1, which can determine the total demand vector X .

$$X = (I - C)^{-1}A. \quad (1)$$

Bandwidth Provision in Cloud. The work in [3–7] implements the techniques to enforce the minimum bandwidth allocation for each VM that is used to host specific services. However, these studies do not consider the policies to determine the appropriate bandwidth capacity for each service and its constituent VMs from a holistic perspective.

The methods proposed in the literature [3,4,8] are mostly job-oriented (or tenants-oriented), i.e., to calculate the resource allocation given the specific tasks submitted by the tenants. However, as we discussed in Sect. 1, service invocations in a service workflow may vary according to the dynamic system information, and therefore it may be difficult to know beforehand the exact execution paths of the workflows in the Cloud. The bandwidth allocation policies developed in this paper are service-oriented, which do not focus on allocating the resources for a set of specific tasks, but aim to allocate the resources for each service based on the interaction patterns among the services.

VMs-to-PMs Placement. Various methods have been developed to address the VM-to-PM mapping problem, including knapsack modelling [9], the mixed integer programming [10], genetic algorithms [11], and heuristic methods [12]. However, the work is used to tackle the placement of independent VMs (i.e., there are no communications among VMs), aiming to minimize the usage of physical machines. In this paper, we investigate the placement method for interacting VMs.

Our previous work in [13] conducted the research in the same problem domain. The work in [13] and this work are in the big scope of the same project on investigating resource management for interacting services in Clouds. Nevertheless, they focus on completely different aspects of the project. The work in [13] focuses on computing resources demanded by services, while this work focuses on the demand for communication resources. The technical contributions in [13] are not attributed to the current work in terms of both developed IO models and VM placement algorithms.

3 Modelling Bandwidth Provision

3.1 The Communication Input-Output Model

We consider a cloud system as an economy, and each service hosted on the cloud as a sector of this economy. Instead of producing goods, cloud services (sectors)

produce and exchange/communicate data over the network. Whereas goods in a real economy are measured by a common currency that is recognised across different sectors, data produced by services is measured in units of bandwidth across the network infrastructure. Similar to the production of goods in an economy as described by Leontief's model, the cause for the production of data by services is also classifiable as internal and external demands.

Internal demand is the data produced by a service as a consequence of a call from another service. Given two services s_i and s_j from service economy S , we define a *consumption coefficient* c_{ij} as Eq. 2, where d_i and d_j denote the average data size produced by s_i and s_j respectively, and p_{ij} denotes the probability that one invocation of s_j causes one invocation of s_i . To understand Eq. 2, suppose s_j is able to produce one unit of data per unit of time (e.g. it is allocated with one unit of bandwidth). Since an invocation of s_j produces d_j amount of data on average, s_j can be invoked $1/d_j$ times in a unit of time, so that the allocated bandwidth (one unit) of s_j is able to transfer the amount of data produced by s_j . As a consequence, the number of invocations to s_i is then given by $(1/d_j)p_{ij}$. Therefore, the total amount of data produced by s_i can be obtained by Eq. 2. As defined by Eq. 2, c_{ij} represents the amount of data produced by service (sector) s_i for each unit of data produced by s_j in a time unit. This is in line with the definition of c_{ij} used in Leontief's model.

$$c_{ij} = \frac{1}{d_j} p_{ji} d_i \quad (2)$$

In contrast, external demand in a cloud economy is the data produced by a service due to the invocation requested by external clients. When a service s_i is at the head of a service workflow (e.g., a login service at the start of a workflow), then the number of times s_i is invoked by the clients in a time unit (which we call the arrival rate of external requests for service s_i and is denoted by λ_i), together with the average amount of data that an invocation of s_i produces (i.e., d_i), determines the amount of data that will be produced by s_i in a time unit due to the external demand. Therefore, the external data demand for s_i , denoted by a_i , can be calculated by

$$a_i = \lambda_i d_i. \quad (3)$$

This definition is also in line with the definition of external demand as defined by Leontief's model. The end clients of the cloud system who trigger service workflows can be regarded as the open sector of the cloud economy which demands data production from the services.

From these derivations, we can see that a cloud economy shares many similar properties to that of a real economy. By Eqs. 2 and 3, we are able to apply the philosophy of Leontief's IO model to a cloud setting as follows.

We denote x_i^{out} as size of data produced by s_i in a time unit in order to meet both internal and external demand (we use "out" to indicate that these are the data that need to be sent out from s_i). We can establish the relation shown in Eq. 4, where X^{out} and A are vectors of dimension $|S|$ holding the data production (x_i^{out}) and external data demands (a_i in Eq. 3) of the Cloud economy,

respectively, and C is the matrix of c_{ij} . Equation 4 establishes the interdependencies within the Cloud economy in terms of data production. x_i^{out} represents the amount of data that may be transmitted over the *uplink* network interface of the PMs that service s_i is hosted in. Note that if s_i and the destination service of some data sent by s_i are located in the same PM, no uplink bandwidth of the PM needs to be consumed for transferring this part of data. In Subsect. 3.2, we will present how to handle this situation and determine the bandwidth allocation for individual VMs that collectively host service s_i .

$$X^{out} = CX^{out} + A \quad (4)$$

In addition to the economy described by Leontief's model, which only considers the amount of goods produced by each sector, we need to calculate the amount of data received by each service in our data demand IO model. This is because Leontief's model does not consider the additional cost associated with a service receiving the data through its host PM's *downlink* network interface.

Among x_i^{out} of data sent by s_i , the amount of $x_i^{out}p_{ij}$ will be sent to s_{ij} . Let c'_{ij} denote the probability that a unit of data produced by s_i is to be received by s_j . Then c'_{ij} can be calculated as $\frac{x_i^{out}p_{ij}}{x_i^{out}} = p_{ij}$. We denote x_{ij}^{out} as the size of data transmitted from s_i to s_j in a time unit and x_{ji}^{in} as the size of data received by s_j from s_i in a time unit, then we have

$$x_{ji}^{in} = x_{ij}^{out} = c'_{ij}x_i^{out}. \quad (5)$$

Additionally, we denote x_i^{in} as the size of data consumed by s_i (i.e., received from all services) in a time unit. x_i^{in} can then be calculated by Eq. 6, where X^{in} is the vector of x_i^{in} and C' is the matrix of c'_{ij} . Equation 6 establishes the relationship between data production (out) and consumption (in).

$$X^{in} = C'X^{out}. \quad (6)$$

3.2 Bandwidth Provision for VMs

From the CIO model, we can derive the amount of data that are communicated by each service. In this section, our objective is to translate this quantity into actual bandwidth provision for individual VMs hosting a service. In a cloud system, each service is hosted by a collection of VMs. We assume that the service is the only service hosted in each of the VMs. This assumption is reasonable since it is a typical setting in Clouds to host different Cloud services in different VMs so as to provide the isolated service environments.

When two VMs of a pair of services are located on the same PM, data may be transmitted locally and thus does not consume the VMs' physical bandwidth. However, in order to take advantage of this local data transmission channel, the local ratio between the numbers of VMs of two service needs to match their global ratio. This is explained in detail below.

Given a pair of services s_i and s_j from S , V_i and V_j denote the total number of VMs in the cloud for hosting these two services, respectively. Consequently,

the amount of data sent from a VM^i (VM^i denotes a VM that hosts service s_i) to service j can be calculated by $\frac{x_{ij}^{out}}{V_i}$, where x_{ij}^{out} is the data sent by service i to j in a time unit, which is calculated by Eq. 4. Given a PM PM_k , v_{ik} and v_{jk} denote the number of VM^i and VM^j in PM_k , respectively. Then in PM_k , the amount of data that are communicated by VM^i 's to service j is $v_{ik} \frac{x_{ij}^{out}}{V_i}$. If $\frac{v_{ik}}{v_{jk}}$ (i.e., the local ratio of the number of VM^i to the number of VM^j in PM_k) is no greater than $\frac{V_i}{V_j}$ (i.e., the global ratio of the number of VM^i to the number of VM^j in the cloud), all data sent by VM^i 's in PM_k (the VMs that host service i in PM_k) to service j can be handled by VM^j 's in PM_k . Therefore, there is no need to consume the bandwidth of VM^i (or VM^j) for sending (or receiving) these data. For example, assume V_i and V_j are 20 and 50, respectively. If in PM_k , v_{ik} is 2 and v_{jk} is 6, then there are more than fair share of VM^j (which is 5) in PM_k to handle the data sent by VM^i in the same machine (since $2/6 < 20/50$).

On the contrary, if the local ratio is greater than the global ratio, which means that there are not adequate VM^j in PM_k to handle the data sent by VM^i in PM_k . The portion of data that cannot be handled by VM^j in PM_k , denoted by y_{ijk} , have to be sent by VM^i to VM^j in another PM, PM_l , and therefore consume the uplink bandwidth of VM^i and the downlink bandwidth of VM^j . y_{ijk} can be calculated by Eq. 7. Equation 7 essentially compares whether the local ratio is no greater than the global ratio. If so, y_{ijk} is 0. Otherwise, Eq. 7 calculates the data that s_i has to send out after deducting the portion of data that can be handled by VM^j in the same machine.

Since y_{ijk} is the data communicated in a time unit, y_{ijk} is essentially the bandwidth that has to be allocated to the VM^i 's in PM_k for sending data to service s_j . Therefore, $\frac{y_{ijk}}{v_{ik}}$ is the uplink bandwidth that has to be allocated to each VM^i in PM_k for sending the data to s_j , while $\frac{y_{ijk}}{v_{jl}}$ is the downlink bandwidth allocated to each VM^j in PM_l for receiving y_{ijk} . The total uplink bandwidth that needs to be allocated to VM^i in PM_k can be calculated by $\sum_{s_j \in PM_k} y_{ijk}$.

$$y_{ijk} = \max\left\{v_{ik} \frac{x_{ij}^{out}}{V_i} \left(1 - \frac{v_{jk}(V_i/V_j)}{v_{ik}}\right), 0\right\} \quad (7)$$

Given a VM-to-PM mapping, denoted by \mathcal{M} , the total uplink communication bandwidth generated by \mathcal{M} can be calculated by Eq. 8, where y_{ijk} is the amount of data that are sent from VM^i (hosting service i) in PM_k (consuming the uplink bandwidth of PM_k) to VM^j (hosting service j) in other PMs. The total downlink bandwidth generated by a VM-to-PM mapping can be calculated in a similar way.

$$\mathcal{C}(\mathcal{M}) = \sum_k \sum_j \sum_i y_{ijk}. \quad (8)$$

4 The Communication-Oriented Simulated Annealing Algorithm

In the classical SA approach, an initial solution is first generated (a solution is encoded) and the neighbourhood searching routine is then applied to generate

new suitable candidate solutions. A cost function and the metropolis criterion [14], which models the transition of a thermodynamic system, are used to determine the quality of the solutions and guide the searching direction so that better solutions can be gradually generated until the stopping criterion is met.

In this section, we design a Communication-oriented SA (CSA) algorithm that aims to find the VM-to-PM mapping with the minimal bandwidth provision for all VMs. In the CSA algorithm, the initial solution is set as the VM-to-PM mapping that is generated by the MinPM algorithm [9] (i.e., the algorithm that produce the VM-to-PM mapping that uses the minimal number of PMs to host VMs). The amount of bandwidth provision calculated in Eq. 8 is used as the cost function for the CSA algorithm. The CSA algorithm adjusts the VM-to-PM mapping, aiming to reduce the bandwidth provision without increasing the number of PMs. This section presents the encoding of the solution, the neighbourhood searching routine and the flow of CSA algorithm in this paper.

Encoding the Solution. In the SA algorithm, a solution is encoded as a two-dimensional array, A , in which an element $a[i][j]$ represents how many VMs of Service s_j there are in PM_i . Note that this encoding method does not differentiate the VMs for the same service. This way, the number of VMs does not affect the complexity of the algorithm. Consequently, the proposed SA algorithm can find the good VM-to-PM mappings efficiently.

Neighbourhood Searching. In SA, the design of neighbourhood searching routine is critical for generating good solutions with good efficiency. This subsection presents the method to conduct the neighbourhood searching. Two probabilities, p_p and p_s , are set to represent the possibility that the VM mapping of a service in a PM is adjusted. To improve the efficiency, the following design is adopted for the neighbourhood searching. The neighbourhood searching routine randomly selects $N \times p_p$ PMs (N is the total number of PMs) to adjust the VM mappings of some services in these PMs. For a selected PM, the routine further randomly selects $M \times p_s$ services (assume M is the number of services in the PM) and the VM mappings of these services will be adjusted. For service s_i in PM_j , its VM mapping is adjusted in the following way. First, the neighbourhood searching routine randomly selects another PM, PM_k , and then randomly selects a service, s_l ($l \neq i$), in PM_k . The routine then tries to swap the VMs between s_i and s_l . In order to render a valid swap, the routine calculates the maximum number of VMs that can be swapped between the two services, which can be calculated using Algorithm 1, where f_k and f_l are the spare resource capacity in PM_k and PM_l , respectively, v_{ik} is the number of VM^i in PM_k , $swap_{ik}$ is the maximum number of VM^i that can be swapped in PM_k . A valid swap is one after which the total capacity of every type of resource (the resource types of CPU utilization, memory and bandwidth are considered in this work) allocated to the VMs in either PM does not exceed the total physical resource capacity of the PM. This validity rule guarantees that the number of required PMs does not increase. The neighbourhood searching is presented in Algorithm 2.

As discussed above, the neighbourhood searching routine randomly selects $N \times p_p$ PMs (N is the total number of PMs) and in each selected PM, the routine

Algorithm 1. Calculating maximum number of VMs that can be swapped

```

1: if  $VM_k^i \times v_{ik} < VM_l^j \times v_{jl}$  then
2:    $Swap_{ik} = v_{ik}$ 
3:    $Swap_{jl} = \lfloor \frac{VM_k^i \times v_{ik} + f_k}{VM_l^j} \rfloor$ 
4: else if  $VM_k^i \times v_{ik} > VM_l^j \times v_{jl}$  then
5:    $Swap_{jl} = v_{jl}$ 
6:    $Swap_{jk} = \lfloor \frac{VM_l^j \times v_{jl} + f_l}{VM_k^i} \rfloor$ 
7: else
8:    $Swap_{ik} = v_{ik}$ 
9:    $Swap_{jl} = v_{jl}$ 

```

Algorithm 2. Neighbourhood searching

```

1: Randomly select  $\lfloor p_p \times N \rfloor$  PMs
2: for each of these PM do
3:   Randomly select  $p_s \times S$   $|_k$  services in  $PM_k$ 
4:   for each of services do
5:     Randomly select a PM,  $PM_l (l \neq k)$  and a service  $j (j \neq i)$  in  $PM_i$ 
6:     Call Algorithm 1 to calculate maximum number of VMs in  $VM^i$  and  $VM^j$  that can form a valid swap
7:     Swap calculated number of VMs between  $s_i$  in  $PM_k$  and  $s_j$  in  $PM_c$ 
8: Return new VM-to-PM mapping,  $\mathcal{M}'$ 

```

further selects $M \times p_s$ services to adjust their VM mappings. Therefore, the time complexity of Algorithm 2 is $O(p_p \times N \times p_s \times M)$.

Simulated Annealing. Algorithm 3 outlines the entire SA process aiming to find the optimal VM-to-PM allocation. In the algorithm, T is the initial temperature of the SA process, which is typically set as 1000. $factor$ is the cool-down factor of the SA process, which is typically set as 0.85. In each iteration, \mathcal{M} is the current VM-to-PM mapping. Algorithm 2 is called to generate a new candidate VM-to-PM mapping, \mathcal{M}' (Line 4). Equation 8 is then applied to calculate the communication cost ($\mathcal{C}'(\mathcal{M}')$) of the new mapping \mathcal{M}' (line 5). If $\mathcal{C}'(\mathcal{M}')$ is better(smaller) than that of the current mapping, the algorithm accepts the new mapping and the new mapping becomes the current mapping (Line 6–8). Otherwise, the metropolis criterion, calculated by $\exp(\frac{-\Delta\mathcal{C}(\mathcal{M})}{T})$, is used to decide whether this new but worse VM mapping should be accepted. If the calculated metropolis criterion is greater than a float number randomly generated between 0 and 1 (Line 7), \mathcal{M}' is accepted. Otherwise, the current mapping remains intact. The iteration repeats until the current mapping stays unchanged for a certain number of consecutive iterations (counted by j) or the number of iterations (counted by i) reaches a pre-set number, k_{max1} and k_{max2} in the CSA (Line 2).

There are at most k_{max2} iterations in the “while” loop in Algorithm 3. In each iteration, calling Algorithm 2 dominates the time spent in an iteration. Therefore, the time complexity of Algorithm 3 is $O(k_{max2} p_p N p_s M)$.

5 Performance Evaluation

We have conducted the simulation experiments to evaluate the effectiveness of the CIO model and the CSA algorithm developed in this work.

The synthetic trace is generated in the simulation experiments. A set of 500 services are generated. A service is defined as the start service, from which all workflows in the trace start. Another service is defined as the end service, which

Algorithm 3. The CSA Algorithm

Require: \mathcal{M}

- 1: $i = 0, j = 0$
- 2: **while** $j \leq k_{max1}$ or $i \leq k_{max2}$ **do**
- 3: $T \leftarrow T \times factor$
- 4: $\mathcal{M}' \leftarrow$ Call Algorithm 2
- 5: $C'(\mathcal{M}') \leftarrow$ Call Eq. 8
- 6: $\Delta C(\mathcal{M}) \leftarrow C'(\mathcal{M}') - C(\mathcal{M})$
- 7: **if** $\Delta C(\mathcal{M}) < 0$ or $\exp(-\frac{\Delta C(\mathcal{M})}{T}) >$
 $R(0, 1)$ **then**
- 8: $\mathcal{M} \leftarrow \mathcal{M}'$
- 9: $j = 0$
- 10: **else**
- 11: $j = j + 1$
- 12: $i = i + 1$

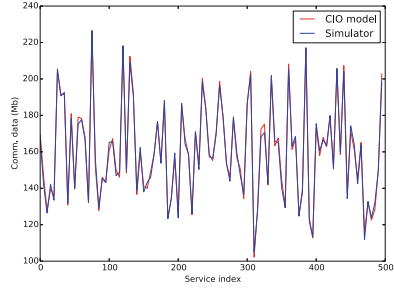


Fig. 1. Accuracy of the CIO model using synthetic traces

means that when the workflow reaches to this service, it will not invoke further services. The degree of parallelism (denoted by DP) is set, which is 3 by default, when generating the workflow instances for the synthetic trace. For all services except the end service, after a service (e.g., s_i) invoked by a task is completed, it further randomly invokes DP (e.g., 3) services. The roulette wheel method is used to randomly determine which DP services are selected based on p_{ij} . In the synthetic trace, the value of p_{ij} is randomly set from the range of $[0.001, 0.003]$ with the average of 0.002 (i.e., $1/500$, where 500 is the number of services generated in the trace). The workflow instance stops growing when all branches in the workflow reach the end service. The technique presented in [13] is used to calculate the number of VMs for each service. The strategy presented in [9] is used to generate the initial VM-to-PM mapping with the minimal number of PMs.

5.1 Accuracy of the CIO Model

It is straightforward to determine p_{ij} for the synthetic trace since a service randomly invokes another service. With p_{ij} , we apply the bandwidth IO model to calculate the bandwidth allocated for each service. In the simulator developed in this work, we allocate the calculated bandwidth to the services and then run the simulation experiments. We record the amount of data that are communicated by each service. If the proposed bandwidth IO model is effective, then the amount of data that are communicated by each service in a time unit in the simulation experiment should equal to the bandwidth allocated to each service. The results are shown in Fig. 1. The average percentage of discrepancy between the CIO model and simulation experiments is 1.3%, which suggests that the CIO model is able to capture the bandwidth demands accurately.

5.2 The Effectiveness of CSA

The experiments in this subsection investigate the effectiveness of the CSA algorithm. In the experiments, we first used the methods proposed in [9], which we call the MinPM algorithm in this paper, to obtain the VM-to-PM mapping that uses the minimal number of PMs to host the VMs. We then apply the proposed SA algorithm to further adjust the VM-to-PM mapping in order to reduce the communication cost without increasing the number of PMs. We also used the greedy method presented in [15] to perform the VM-to-PM mapping and compared the results against those generated by the proposed SA. In the greedy algorithm, all services are ranked in the decreasing order of their communication intensity (i.e., the data that have to be communicated by a service in this paper). The greedy algorithm first place the VMs of the first service (i.e., the one with most communication intensity) on PMs, with each PM having the same number of VMs or having at most ± 1 difference if it can not be evenly divided). Then the greedy algorithm selects the next service, s_2 , and tries to place its VMs to PMs so that the local ratio of the number of VMs of s_1 to that of s_2 in a PM equal (or is the closest) to the global ratio of the total number of VMs of s_1 to that of s_2 . The procedure repeats until all VMs are mapped.

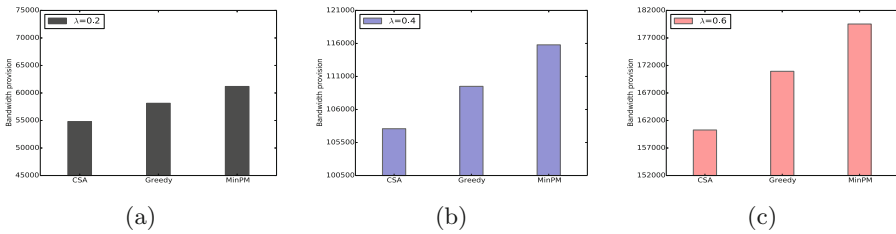


Fig. 2. Comparing CSA with other existing algorithms using synthetic trace

We increase the arrival rate of the generated workflows and use the technique presented in [13] to calculate the number of VMs for each service under different arrival rates. The experimental results are shown in Fig. 2(a, b and c). It can be seen that CSA outperforms other two algorithms in all cases.

References

1. Amazon case study: Nasaq OMX. <http://goo.gl/28wfvG>
2. Leontief, W.: Input-output analysis. New Palgrave Dictionary of Economics (1987)
3. Jalaparti, V., Ballani, H., Costa, P., Karagiannis, T., Rowstron, A.: Bridging the tenant-provider gap in cloud services. In: ACM SOCC (2012)
4. Meng, X., Pappas, V., Zhang, L.: Improving the scalability of data center networks with traffic-aware virtual machine placement. In: IEEE INFOCOM (2010)
5. Popa, L., Kumar, G., Chowdhury, M., Krishnam. A., Ratnas, S., Stoica, I.: Fair-cloud: sharing the network in cloud computing. In: ACM SIGCOMM (2012)

6. Ballani, H., Costa, P., Karagiannis, T., Rowstron, A.: Towards predictable data-center networks. In: ACM SIGCOMM (2011)
7. Ballani, H., Jang, K., et al.: Chatty tenants and the cloud network sharing problem. In: Proceedings of NSDI2013 (2013)
8. Jiang, J.W., Lan, T., et al.: Joint VM placement and routing for data center traffic engineering. In: IEEE INFOCOM (2012)
9. Hermenier, F., Lorca, X., Menaud, J.-M., Muller, G., Lawall, J.: Entropy: a consolidation manager for clusters. In: 2009 ACM SIGPLAN/SIGOPS (2009)
10. Petrucci, V., et al.: A dynamic optimization model for power and performance management of virtualized clusters. In: Proceedings of e-Energy 2010 (2010)
11. He, L., Zou, D., et al.: Developing resource consolidation frameworks for moldable virtual machines in clouds. *Future Gener. Comput. Syst.* **32**(1), 69–81 (2013)
12. Hu, L., Jin, H., Liao, X., Xiong, X., Liu, H.: Magnet: a novel scheduling policy for power reduction in cluster with virtual machines. In: *Cluster* (2008)
13. Chen, C., He, L., Chen, H., Sun, J., Gao, B., Jarvis, S.: Developing communication-aware service placement frameworks in the cloud economy. In: *Cluster* (2013)
14. Van Laarhoven, P.J., Aarts, E.H.: *Simulated Annealing*. Springer, Heidelberg (1987)
15. He, L., Jarvis, S.A., Spooner, D.P., Jiang, H., Dill, D.N., Nudd, G.R.: Allocating non-real-time and soft real-time jobs in multiclusters. In: *TPDS* (2006)

Four-Fold Auto-Scaling on a Contemporary Deployment Platform Using Docker Containers

Philipp Hoenisch^{1,2(✉)}, Ingo Weber^{2,3}, Stefan Schulte¹, Liming Zhu^{2,3},
and Alan Fekete^{2,4}

¹ TU Wien, Vienna, Austria

`{p.hoenisch,s.schulte}@infosys.tuwien.ac.at`

² Software Systems Research Group, NICTA, Sydney, Australia

`{Ingo.Weber,Liming.Zhu,Alan.Fekete}@nicta.com.au`

³ School of Computer Science and Engineering, University of New South Wales,
Sydney, Australia

⁴ School of Information Technologies, University of Sydney, Sydney, Australia

Abstract. With the advent of Docker, it becomes popular to bundle Web applications (apps) and their libraries into lightweight linux containers and offer them to a wide public by deploying them in the cloud. Compared to previous approaches, like deploying apps in cloud-provided virtual machines (VMs), the use of containers allows faster start-up and less overhead. However, having containers inside VMs makes the decision about elastic scaling more flexible but also more complex. In this contemporary approach to service provisioning, four dimensions of scaling have to be considered: VMs and containers can be adjusted horizontally (changes in the number of instances) and vertically (changes in the computational resources available to instances). In this paper, we address this four-fold auto-scaling by formulating the scaling decision as a multi-objective optimization problem. We evaluate our approach with realistic apps, and show that using our approach we can reduce the average cost per request by about 20–28 %.

1 Introduction

With the advent of Docker¹, lightweight containers are gaining wide-spread popularity among early adopter-type companies, including Facebook and Google [15]. Such containers are also particularly suitable to power recent trends like microservices and DevOps [1]. We consider the approach where a variety of different services or Web applications (apps) are running inside containers, each app deployed in instances of a particular container type. Various container instances are deployed on top of VM instances, which can be obtained from Infrastructure-as-a-Service (IaaS) providers, e.g., Amazon Web Services (AWS).

Virtualization brought the benefit of detaching the setup of a machine from a physical machine (PM), and enabled IaaS cloud offers. VMs contain a full operating system (OS) and all software required for a specific purpose. Further, a

¹ <http://docker.io>, accessed 29/7/15.

VM's configuration can be packaged into an image and cloned arbitrarily often. This mechanism is commonly used for apps running in the cloud [19]. Most recently, the idea of having an additional abstraction layer has been picked up by lightweight containers which can be described as *smaller* variants of VMs: the OS is not included in a container; instead, the one from the host machine is used. However, most additional programs are included inside the container. Like VMs, containers offer *resource elasticity, isolation, flexibility* and *dependability*. Containers do need to run on compatible versions of the OS, and they share resources through the outside OS, but these limitations are compensated by benefits of faster start-up times (on the order of seconds, where booting a VM takes on the order of minutes) and less overhead in terms of used resources (since the containers do not require a separate OS). One can certainly use Docker containers (from now on only called *containers*)² *instead of* VMs directly on an OS that runs on private PMs. However, in public cloud platforms such as AWS, the norm is running containers *inside* VMs obtained from the IaaS providers. These VMs are initially launched with only the OS and no app-specific software. That raises a complicated question of allocation: *how many VMs are needed, and how should the containers be configured and distributed among the VMs?* Compared to instance allocation of VMs in former approaches, container deployment requires making allocation decisions from a much larger space of options. As in all cloud platforms, allocation needs to be dynamic and elastic: as the load on an application changes, the amount of resources used should be adjusted to match the change, so the costs can be scaled up and down following usage.

In this paper we propose to use *a multi-objective optimization model to solve the allocation problem for lightweight containers on top of VMs*. In this model, we consider horizontal and vertical auto-scaling on both the container and VM level. Hence, we make the following contributions:

- We offer a multi-objective optimization model for scaling a contemporary deployment platform (consisting of containers and VMs) including the interactions between the different layers. This has more degrees of freedom than scaling just VMs as in traditional cloud deployment scenarios.
- We show how the optimization model can be used for making auto-scaling decisions for this deployment platform. The control decisions require solving the optimization problem, which we can do for reasonably-sized distributed systems, even though the additional degrees of freedom make the optimization task more complex than for traditional approaches.
- Using a prototype, we provide a realistic evaluation of our approach against two baseline scenarios which consider fewer dimensions of scaling. In our experiments we achieved total savings of up to 28%.

The remainder of the paper is structured as follows. We start with a motivating example and give an overview of our approach in Sect. 2. In Sect. 3 we present the

² In our current implementation we use Docker as container technology. The approach however would work for any lightweight container supporting resource and application isolation, scalability and dependability.

details of the optimization problem. We discuss the results of our experimental evaluation in Sect. 4 and related work in Sect. 5. Section 6 concludes the paper. An accompanying technical report (TR) [8] provides details that were omitted here for brevity.

2 Motivating Example

Consider the Following a Platform-as-a-Service (PaaS) Scenario: we assume the role of a provider hosting various apps using a public IaaS cloud service³. Apps have a specific *type* and are provided by different customers or content providers who want to have a fixed hosted solution. Each app may come with an optional *Service Level Agreement* (SLA), e.g., defining a maximal *response time* which should not be exceeded or a specific *throughput* which should be achieved. For the sake of simplicity, say the provider hosts the following three apps for three customers, all subject to varying workloads. *App A*: Joomla with extensions for appointment management, for customer 1, a hairdresser; *App B*: Wordpress with plugins for CRM and Lead Management, for customer 2, a tech startup; and *App C*: NodeJS with the web site of customer 3, a skiing tour operator. As these apps would interfere with each other, each app is packaged into a separate container type. For that, a *dockerfile* specifies the configuration necessary to start containers, i.e., a new *container instance* of the same container type. New containers for the apps can then be instantiated as desired. Common practice is to specify version and build numbers or commit hashes in the dockerfile, so that all containers spawned from the same dockerfile start off as being the same, i.e., running the same application code, versions and libraries. If an app needs to be upgraded, this is initiated by changing the dockerfile.

The provider leases *VMs* of different *types*, i.e., each VM type has a different configuration in terms of supplied resources. In order to deploy containers onto a VM, a VM needs to be instantiated resulting in a VM *instance*. Notably, the provider can deploy many different containers (of different types) on a single VM instance. In addition, the provider may deploy a specific container type on several VM instances, resulting in various container instances where each container instance may have a different *configuration*. This configuration can be used to ensure the app's SLA is met, by defining requirements on the underlying VM, e.g., the resource demand in terms of CPU and RAM. The requirements on CPU are defined in *CPU shares*. By default, each VM has 1024 CPU shares available which are split between the hosted container instances. Say the provider leases a dual-core VM for all three apps and the respective numbers of requests rise during peak times – therefore the system needs to be scaled. The problem then is: *how many VMs of which types are needed, and how should the containers for the different apps be distributed among them, with which configuration?*

³ Internal IT operation units face a very similar situation when providing container hosting to their organization. Hence, instead of a public cloud IaaS a private cloud can be used.

If the demand for all three apps increases more or less uniformly, it may be sufficient to lease more VMs, each hosting all three apps. However, if the demand for App A grows a lot faster than for App B and C, more resources need to be provisioned for it. If later App A’s demand shrinks, the freed up resources may be released. Alternatively, if App B experiences increased demand at that point, the resources freed up by App A can be re-purposed for App B.

As can be seen, the decision of how many VMs of which type should be leased when and how to distribute and configure the containers is not straight forward. Hence, in this paper, we make use of an optimization approach and define four-fold auto-scaling as a multi-objective optimization model.

3 Optimization Approach

Building on the example scenario and the main aspects of our problem landscape, we now give an overview of our multi-objective optimization model. Due to space constraints, we present only the main objective function and refer to our technical report (TR) for more information [8]. The optimization model takes as input a set of different VM types ($V = \{1, \dots, v^\#\}$) and container types ($D = \{1, \dots, d^\#\}$). v corresponds to a VM type and k_v to a specific VM instance. Accordingly, d refers to a container type and c_d to a specific container instance (having a certain configuration). Each VM instance and container instance comprises a certain specification in terms of CPU and RAM. For the sake of simplicity, we generalize all types of resources here; in our implementation we differentiate CPU shares and RAM.

The goal of the objective function below is minimizing the overall cost, i.e., the cost for all leased VMs. Hence, the output of the model is two-fold: first it defines how many VM instances of which types are needed, and second, which container (including its configuration) should be deployed on which VM instance. The decision variable $x_{(c_d, k_v, t)}$ is set by the solver and defines which container c_d should be deployed on which VM instance k_v at time t .

$$\min \left[\sum_{v \in V} c_v \cdot \gamma_{(v, t)} + \sum_{d \in D} \sum_{c_d \in C_d} \sum_{v \in V} \sum_{k_v \in K_v} ((1 - z_{(d, k_v, t)}) \cdot (x_{(c_d, k_v, t)} \cdot \Delta_d)) \right. \\ \left. + \sum_{v \in V} \sum_{k_v \in K_v} \omega_f^R \cdot f_{(R, k_v, t)} + \sum_{d \in D} \sum_{c_d \in C_d} \sum_{v \in V} \sum_{k_v \in K_v} (\omega_s \cdot s_{(i, c_d, t)} \cdot x_{(c_d, k_v, t)}) \right]$$

The objective function comprises four terms. The first term $\sum_{v \in V} c_v \cdot \gamma_{(v, t)}$ computes the overall VM leasing cost: $\gamma_{(v, t)}$ many VM instances of type v with cost c_v are leased at time t . The second term $\sum_{d \in D} \sum_{c_d \in C_d} \sum_{v \in V} \sum_{k_v \in K_v} ((1 - z_{(d, k_v, t)}) \cdot (x_{(c_d, k_v, t)} \cdot \Delta_d))$ sums up the time needed to deploy a container (Δ_d) on a VM instance k_v . If a specific container of type c_d gets deployed the first time on a VM instance k_v , some data needs to be downloaded from the container registry. Hence, this procedure may take some time. However, this data is cached on the VM instance as long as it is running, thus future deployments

of the same container type will be much faster. In case a VM instance’s cache contains already the needed data ($z_{(d,k_v,t)}=1$), the product inside the sums is 0. Further, this term prioritizes placement of containers on VM instances where they are cached already, since the term is minimized as part of the overall objective function. The third term $\sum_{v \in V} \sum_{k_v \in K_v} (\omega_f^R \cdot f_{(R,k_v,t)})$ computes the amount of *free* resources ($f_{(R,k_v,t)}$). This term ensures that containers are deployed on already leased VM instances instead of leasing additional ones, provided enough resources are available. In order to control the contribution of this term towards the objective function, we weigh the free resources using the weight ω_f^R . The fourth term $\sum_{d \in D} \sum_{c_d \in C_d} \sum_{v \in V} \sum_{k_v \in K_v} (\omega_s \cdot s_{(i,c_d,t)} \cdot x_{(c_d,k_v,t)})$ sums up the amount of deployed containers for each container type at time t . It aims at avoiding over-provisioning on the container level by demanding to lease the smallest amount of resources to containers while still fulfilling the demand. As before, the term is weighed with a constant value ω_s .

The full optimization model can be found in the TR [8]. It includes numerous constraints, which define limits on valid container deployments and VM leasing plans. Overall, the optimization model ensures that enough resources are leased to handle the demand for each app at any time. Hence, the outcome of the optimization model is two-fold: (1) it determines whether additional VM instances need to be leased or if already leased VM instances can be terminated, and (2) it determines which container types to instantiate on which VM instance. Consequently, the system landscape subject to continuous change: VMs may appear or disappear at any time and containers may be moved between them.

4 Evaluation

Our optimization-based control of auto-scaling has been evaluated through measuring the behavior of a prototype. The source code is available at <http://reliableops.com>. Details of the architecture and evaluation can be found in the TR [8]. We compare our *optimized* approach against two state-of-the-art baselines for making scaling decisions: *One-for-All* and *One-for-Each*. In both cases, additional resources are leased or released based on a threshold. *One-for-All* leases only quad-core VMs, where each VM hosts one container of each type. *One-for-Each* leases only single-core VMs, each hosting exactly one container. As in the example scenario in Sect. 2, we have three different container types (i.e., three apps). We test using two request arrival patterns, sending different and varying amounts of requests to each app.

Table 1. Evaluation results

	Arrival Pattern 1			Arrival Pattern 2		
	Optimized	One-for-All	One-for-Each	Optimized	One-for-All	One-for-Each
Leased cores (σ)	28.13 (0.38)	39.5 (0)	29.68 (0.14)	28.75 (0.43)	39.5 (0)	29.88 (0.25)
Leasing cost (σ)	378.4 (0.92)	505.6 (0)	474.67 (2.31)	393.2 (1.38)	505.6 (0)	478.00 (4)
Cost/Invocations (σ)	0.17 (0.012)	0.23 (0.22)	0.21 (0.02)	0.17 (0.03)	0.24 (0.06)	0.22 (0.04)
SLA adherence	97.47 %	98.02 %	98.22 %	96.95 %	96.92 %	97.1 %

The result of our evaluation are shown in Table 1. As the numbers reveal, the SLA adherence for both arrival patterns and for each scaling strategy are very similar, varying by less than 2.5%. This can be expected given that we used the same thresholds for scaling up or down. Hence, in the following we focus on the leased CPU cores and incurred cost. It is not surprising that the *One-for-All* scenario produced the highest cost. In this scaling strategy, only quad-core VMs were leased, each hosting one container instance per app. This leads to a highly over-provisioned system when some apps experience relatively low load, as VMs are not be fully used in this case. Hence, leasing single-core VMs with only one app as in *One-for-Each* is $\sim 5\%$ cheaper. However, even leasing smaller VMs may not be perfect. Since single-core VMs can not service as much load as more powerful VMs, more VMs are needed. In addition, in our cost model, leasing two single-core VMs is more expensive than leasing one dual-core VM ($\sim 20\%$ more expensive). Thus vertical scaling can be helpful: leasing the right VM size depending on the need may eventually lead to less leasing cost. Our four-fold optimization approach can take advantage of this, as can be seen in the *Cost/Invocations* row in Table 1: Eventually, for *Arrival Pattern 1*, we achieved savings of $\sim 28\%$ (with $\sim 33\%$ less cores) over the One-for-All scenario and monetary savings of $\sim 23\%$ ($\sim 4\%$ less cores) over the One-for-Each. For *Arrival Pattern 2*, we achieved with our optimization cost savings of $\sim 25\%$ ($\sim 32\%$ less cores) over the One-for-All scenario and savings of $\sim 20\%$ ($\sim 4\%$ less cores) over the One-for-Each scenario.

5 Related Work

Resource allocation and auto-scaling are major research challenges in the field of cloud computing [3]. Several different approaches have been proposed for scaling single services, scientific workflows and business processes. Approaches for both fields differ in a number of aspects, e.g., process perspective, timeliness, resource insensitivity, scheduling on step-level or for the full process, etc. [6, 7, 16, 17]. The major difference of process-based scaling versus scaling single services lays in the versatility, i.e., the amount of parallel requests may jump within a few seconds from a low to a very high number. Hence, the demand of resources may also change quickly which makes scaling decisions more complicated [2]. As resources are commonly paid for a fixed Billing Time Unit (BTU) (several minutes to a few hours), releasing resources before the end of such a cycle should be avoided.

The adherence to SLAs has also been investigated. SLAs are defined by customers under the objective of optimizing profit for the IaaS provider [11], or to achieve a high resource utilization [5, 9, 10]. For that, most approaches apply threshold-based scaling, i.e., fixed rules apply depending on the current load. Li et al. propose using reinforcement learning to reason about the best configuration to ensure a certain level of QoS [12]. However, it is more realistically for service providers to isolate different apps: they may have conflicts amongst each other, like exposing the same ports, using incompatible libraries, or privacy constraints preventing their deployment in the same VM [18].

To overcome this configuration problem, commonly apps and their libraries are packed into a single VM. Doing so, a more fine grained scaling is possible. However, from the point of view of a IaaS provider, the problem of unused resources has only been shifted. Now the question is how to use the PMs efficiently. A number of solutions have been proposed which consider different VM placement approaches with the aim of utilizing the physical resources efficiently [4, 14, 20]. As it is common that apps communicate with each other, having them in separate VMs will eventually lead to high data transfer. Cloud providers often charge for in and out-bound traffic. Hence, VM placement should consider a collocation of VMs which communicate regularly with each other [6, 13]. Approaches for scaling VMs and placing them on PMs are particularly relevant to our work. However, using VMs instead of containers should be limited to cases where a full OS is needed while containers should be used to isolate single apps. Further, optimizing VM placement is only applicable for datacenter operators with hardware access. Where this is not the case, using containers in combination with VMs allows much finer-grained scaling and resource usage optimization.

6 Conclusions

The traditional approach of hosting apps directly on VMs suffers from several disadvantages, such as the overhead of a full OS and slow start-up time, a degree of vendor lock-in, and relatively coarse-grained units for scaling. In order to overcome those problems more and more organizations use lightweight container technologies like Docker. These add an additional abstraction layer on top of VMs, enabling more efficient use of VMs. When running containers on top of VMs, auto-scaling decisions have greater flexibility and greater complexity. In this paper, we defined a four-fold auto-scaling decision problem for this deployment scenario as a *multi-objective optimization model*, and we proposed a control architecture that dynamically and elastically adjusts VM and container provisioning. Based on a prototype implementation, which uses IBM CPLEX as optimization solver, we evaluated our approach extensively, comparing it against two naïve scaling strategies. We showed that our approach can choose and execute scaling decisions, achieving a cost reduction of 20–28% over the baselines. In our future work we want to extend this optimization model and consider the location of containers, privacy aspects, and long-running transactions.

Acknowledgments. We thank An Binh Tran for sharing his technical expertise on Docker, and IBM for the academic license of CPLEX. NICTA is funded by the Australian Government as represented by the Department of Communications and the Australian Research Council through the ICT Centre of Excellence program. This work is partially supported by the European Union within the SIMPLI-CITY FP7-ICT project (Grant agreement no. 318201) and within the CREMA H2020-RIA project (Grant agreement no. 637066).

References

1. Bass, L., Weber, I., Zhu, L.: *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional, Boston (2015)
2. Brousse, N.: Scaling on EC2 in a fast-paced environment. In: *USENIX LISA* (2011)
3. Buyya, R., Ranjan, R., Calheiros, R.N.: InterCloud: utility-oriented federation of cloud computing environments for scaling of application services. In: Park, J.H., Hsu, C.-H., Yeo, S.-S., Yang, L.T. (eds.) *ICA3PP 2010, Part I*. LNCS, vol. 6081, pp. 13–31. Springer, Heidelberg (2010)
4. Chen, W., Qiao, X., Wei, J., Huang, T.: A profit-aware virtual machine deployment optimization framework for cloud platform providers. In: *IEEE CLOUD* (2012)
5. Emeakaroha, V.C., Brandic, I., Maurer, M., Breskovic, I.: SLA-aware application deployment and resource allocation in clouds. In: *COMPSAC Workshops* (2011)
6. Hoenisch, P., Hochreiner, C., Schuller, D., Schulte, S., Mendling, J., Dustdar, S.: Cost-efficient scheduling of elastic processes in hybrid clouds. In: *IEEE CLOUD* (2015)
7. Hoenisch, P., Schuller, D., Schulte, S., Hochreiner, C., Dustdar, S.: Optimization of complex elastic processes. *IEEE Transactions on Services Computing* (2015)
8. Hoenisch, P., Weber, I., Schulte, S., Zhu, L., Fekete, A.: Four-fold auto-scaling for Docker containers. Technical report, UNSW-CSE-TR-201513, University of New South Wales (2015). <ftp://ftp.cse.unsw.edu.au/pub/doc/papers/UNSW/201513.pdf>
9. Juhnke, E., Dörnemann, T., Bock, D., Freisleben, B.: Multi-objective scheduling of BPEL workflows in geographically distributed clouds. In: *IEEE CLOUD* (2011)
10. Kertesz, A., Kecskemeti, G., Brandic, I.: An interoperable and self-adaptive approach for SLA-based service virtualization in heterogeneous cloud environments. *Future Gener. Comput. Syst.* **32**, 54–68 (2012)
11. Lee, Y.C., Wang, C., Zomaya, A.Y., Zhou, B.B.: Profit-driven service request scheduling in clouds. In: *IEEE CCGrid* (2010)
12. Li, H., Venugopal, S.: Using reinforcement learning for controlling an elastic web application hosting platform. In: *IEEE ICAC 2011* (2011)
13. Meng, X., Pappas, V., Zhang, L.: Improving the scalability of data center networks with traffic-aware virtual machine placement. In: *IEEE INFOCOM* (2010)
14. Mills, K., Filliben, J., Dabrowski, C.: Comparing VM-placement algorithms for on-demand clouds. In: *IEEE CloudCom* (2011)
15. MSV, J., McCrory, C.: Is docker a threat to the cloud ecosystem?, August 2014. <http://research.gigaom.com/2014/08/is-docker-a-threat-to-the-cloud-ecosystem/>. Gigaom Research
16. Schulte, S., Janiesch, C., Venugopal, S., Weber, I., Hoenisch, P.: Elastic business process management: state of the art and open challenges for BPM in the cloud. *Future Gener. Comput. Syst.* **46**, 36–50 (2015)
17. Schulte, S., Schuller, D., Hoenisch, P., Lampe, U., Dustdar, S., Steinmetz, R.: Cost-driven optimization of cloud resource allocation for elastic processes. *Int. J. Cloud Comput.* **1**(2), 1–14 (2013)
18. Shen, Z., Subbiah, S., Gu, X., Wilkes, J.: Cloudscale: elastic resource scaling for multi-tenant cloud systems. In: *ACM SOCC* (2011)
19. Varia, J.: *Architecting for the cloud: best practices*. Amazon Web Services Whitepaper (2011)
20. Xu, J., Fortes, J.: Multi-objective virtual machine placement in virtualized data center environments. In: *GreenCom - CPSCOM* (2010)

An SLA-Based Advisor for Placement of HPC Jobs on Hybrid Clouds

Kiran Mantripragada, Leonardo P. Tizzei, Alecio P.D. Binotto,
and Marco A.S. Netto^(✉)

IBM Research, Sao paulo, Brazil
{kiran,ltizzei,abinotto,mstelmar}@br.ibm.com

Abstract. Several scientific and industry applications require High Performance Computing (HPC) resources to process and/or simulate complex models. Not long ago, companies, research institutes, and universities used to acquire and maintain on-premise computer clusters; but, recently, cloud computing has emerged as an alternative for a subset of HPC applications. This poses a challenge to end-users, who have to decide where to run their jobs: on local clusters or burst to a remote cloud service provider. While current research on HPC cloud has focused on comparing performance of on-premise clusters against cloud resources, we build on top of existing efforts and introduce an advisory service to help users make this decision considering the trade-offs of resource costs, performance, and availability on hybrid clouds. We evaluated our service using a real test-bed with a seismic processing application based on Full Waveform Inversion; a technique used by geophysicists in the oil & gas industry and earthquake prediction. We also discuss how the advisor can be used for other applications and highlight the main lessons learned constructing this service to reduce costs and turnaround times.

1 Introduction

Current work on HPC cloud has focused on understanding the cost-benefits of cloud over on-premise clusters [1, 6–9, 13, 15–17, 20, 23]. However, there is still a gap between this understanding and helping users make decisions on bursting their jobs to the cloud. While applications may suffer network overhead, cloud is more effective when we consider resource availability—users do not have to wait long time periods in job queues of cluster management systems.

This paper introduces an advisory service to support users in deciding how to distribute computing jobs between on-premise and cloud resources. Our main contributions are: (i) an advisory service for bursting jobs to the cloud, considering performance and cost difference between cloud and on-premise resources, as well as deadline, local job queue, and application characteristics (Sect. 2); (ii) a case study that shows the advisory service being used by a seismic processing application from the oil & gas industry. We also measured the impact of unreliable execution time predictions on cloud bursting decisions (Sects. 3 and 4).

Extended version of this paper is available at arxiv.org.

2 Advisory Service and Policies

The advisory service considers a user deadline, incurred costs, the on-premise job queue length (local), the provisioning time (cloud), the price ratio between local and cloud for the resource allocation, the type of available hardware, and the estimated execution time for both environments with different configurations.

The main input parameters to the advisor are the *application profiles* and the *cost models*. The *application profiler* generates profiles that describe the behavior of a given application considering infrastructure, financial costs, performance, and number of required processors. Several approaches exist to produce application profiles [2, 3, 10, 18, 22, 24]. The *cost model* for a cloud infrastructure comes from price values offered by cloud providers, whereas the model for the on-premise cluster is a ratio based on the cloud costs [14].

The advisory service currently supports two policies: (i) the maximum *budget* for running jobs, when users are more concerned about costs; (ii) the maximum *execution time*, when users must meet a deadline to deliver results, and budget is a secondary concern. Both policies readjust the number of cores for the cloud environment due to restrictions in the number of cores per machine.

3 Application Case Study in Oil and Gas Industry

The case study of our advisory service relies on the application profile for the Full Waveform Inversion (FWI) [21] and cost models for the SoftLayer cloud provider and an on-premise cluster. FWI is a CPU-intensive application in the area of seismic analysis that has components present in other HPC applications, such as communication among multiple processes, solvers for linear systems, matrix operations, among others. We assume that the profile of such applications can be represented by a power-law function due to its inherent scale-invariance characteristics. Hence, similar applications can be scaled to a finer or coarser grid resolution and will behave similarly, by simple tuning the coefficients of the power-law function [11, 12]:

$$t = aP^b \quad (1)$$

where t is the execution time, P is the number of processors, and the coefficients a and b are empirically determined. We can also invert Eq. 1 to solve the number of processors for a given time restriction t as the input parameter.

In order to develop a *cost model*, we collected prices charged by cloud providers; in our case, SoftLayer¹ cloud infrastructure. Similar findings from our experiments could be obtained using other cloud providers as they rely on similar prices and charging models (hourly-based). Heterogeneity [5] will be explored as future work. We observed that the hourly-rate for provisioning nodes is a linear relationship to the number of processors P , which can be described by ($C_h = \alpha P + \beta$). For simplification purposes, we assume the offset coefficient (β) can be neglected and the “price per hour”:

¹ SoftLayer website: <http://www.softlayer.com/>.

$$\frac{\Delta C}{\Delta t} = \alpha P \quad (2)$$

where $\frac{\Delta C}{\Delta t}$ is the hourly-rate for nodes provisioning and α is a linear coefficient determined empirically.

We can integrate Eq. 2 over time to quantify the total cost for a given turnaround time (the number of processors P does not change with time), while we simplified the costs of on-premise HPC clusters by assuming it is proportional to cloud costs: [8, 14]: $C_{cloud} = T(\alpha P)$ and $C_{local} = T(K\alpha P)$, where T is the turnaround time and C is the total cost for a given number of processors P and turnaround time T . By coupling the application and costs models (Eqs. 1 and 2), we can have one equation that provides C (total cost) for a given time T , a cost model coefficient (α), and the application profile (from a and b):

$$C = a\alpha \left[\frac{\left(\frac{T}{a}\right)^{\left(1+\frac{1}{b}\right)}}{1 + \frac{1}{b}} \right]. \quad (3)$$

4 Evaluation

The goals of the evaluation are to understand: (i) the financial and time savings of the advisor and (ii) how the advisor is dependent from the application profile accuracy. We compared the advisor against four policies:

- **Always-Local:** submits jobs to the on-premise environment—it represents users who do not want or cannot move their jobs to the cloud. We used this policy as baseline for comparison because it still represents the most conservative and traditional behavior of HPC users;
- **Always-Cloud:** submits jobs to the cloud—it represents users who do not have access to an on-premise cluster or are willing to test the cloud to avoid acquiring a new cluster in the future;
- **Random:** randomly decides between cloud and local environments—it is an attempt to represent users who do not have any supporting mechanism or intuition to know where to run their jobs;
- **Worst-Case:** chooses the opposite environment provided by the advisor—it represents hypothetical users who make extremely wrong decisions. This helps us understand how much a user can loose with such decisions.

Other policies [4, 19] could be studied, however finding the optimal resource allocation policy is out of the scope of this paper.

The input data were: deadline ranges from 1 to 100 h; budget ranges from 10 to 100 USD; queue size time ranges from 1% to 50% of the deadline; setup time ranges from 1% to 50% of the deadline; price ratio between cloud and local environments, with the price of cloud environment ranging from 70% to 340% the price of the local environment; total of 28,000 executions per policy. The ranges for the budget, deadline, and setup time are based on our experience

with the FWI application. The price ratio is based on HPC cloud literature [8]. The FWI profile was generated using a single input data set, which described the size of the domain, the precision of the output image, and the varying number of processors from 10 to 40.

The advisor computes the costs and turnaround time for both environments for each set of input variables. For each result, the advisor calculates the relative difference between the costs of both environments in the following way:

$$\frac{\min(Cost_{cloud}, Cost_{local}) - Cost_{local}}{Cost_{local}} \quad (4)$$

where $Cost_{local} > 0$. When the budget-aware policy is executed, the advisor calculates the relative difference of the turnaround time in a similar manner. The results of the other decision policies used for comparison are also relative to the always-local decision policy.

We selected two environments to compare the target application. Cloud: processor frequency = 2.60 GHz, cores per processor 4, memory per machine 64 GB, Ethernet network, CentOS operating system. Cluster: processor frequency = 2.80 GHz, cores per processor 10, memory per machine 132 GB, Ethernet/ Infiniband network, RHEL operating system.

Application profiles describe resource consumption of the application. We derived the power-law scaling function (Eq. 1), in which the coefficients a and b were computed through non-linear least squares curve fitting:

$$t_{local} = 1013.50 P_{local}^{-1.58} \text{ and } t_{cloud} = 7004.86 P_{cloud}^{-2.06}.$$

4.1 Results: Costs and Time Savings

Figure 1 shows the results for the deadline-aware policy. When local environment is cheaper ($K < 1$) and even 80% ($K = 1.8$) more expensive than cloud, it delivers the cheapest cost most of the time. When the price ratio is 1.8, local environment provides the cheapest cost around 71% of the instances. The reason for this is that local environment showed the best computing performance for executing the target application. Thus, even when the local environment is around 80% more expensive than cloud, its performance counterbalances its cost. When the price ratio reaches 2.2, the local environment is surpassed by cloud in terms of cost, *i.e.*, cloud is the cheapest environment around 56% of the time and this percentage becomes greater as the price ratio grows, as expected.

Figure 2 shows the results for the budget-aware policy. Similarly to deadline-aware, always-cloud is comparable to worst-case and advisor is comparable to always-local when the local price is equal to or below the cloud price. However, for higher price ratios, always-cloud surpasses always-local faster than deadline-aware. The turning point occurs when price ratio is around 1.0: always-local is on average 30% faster than always-cloud, but the median is -0.12 ; that is, always-cloud is half the time at least 12% faster than always-local. Although the local environment has better computing performance results over cloud, when they have a similar price (*i.e.*, price ratio around 1.0), the decision on where to run for

a given budget is not obvious due to the other input parameters (queue length and setup time), affecting the turnaround time.

When the advisor calculates an execution time to meet the budget, the coupled model yields a solution that is near-optimal for execution time. Searching for the optimal execution time is not worthwhile since the adjustments over the infrastructure to meet the available configurations overpass intermediate values found in the optimal solution. For instance, an estimated number of processors $NProcs = 9.5$ must be adjusted to 10 or 9, according to the policies.

Results show the advisor selects the environment that best suits users' needs. The lack of such supporting tool might cause unnecessary costs or waste of time. Besides, some input variables (*e.g.*, queue size time) can change frequently, making impossible to manually calculate the best environment for execution.

4.2 Results: Accuracy of the Application Profile

We defined the range of inaccuracies from -90% (*i.e.*, -0.9) error to 100% (*i.e.*, 1.0) error, aiming to cover a wide spectrum of such profiles. For each set of input data, the application profile specifies the infrastructure necessary to meet deadline or budget constraints depending on the policy. Let us say this infrastructure has 100 cores disregarding the policy; after injecting the error within the aforementioned range, it will have from 10 (*i.e.*, $100 * (1.0 - 0.9)$) to 200 cores (*i.e.*, $100 * (1.0 + 1.0)$).

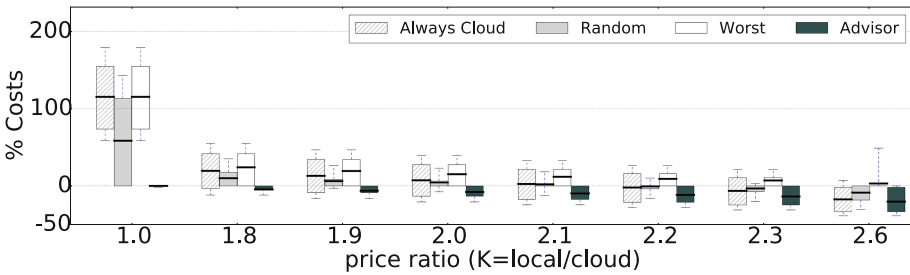


Fig. 1. Boxplots of the Deadline-aware policy: the lower the costs the better the policy

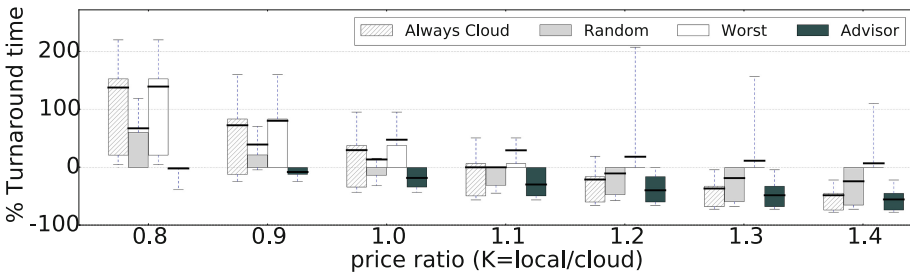


Fig. 2. Boxplots of the Budget-aware policy: the lower the time, the better the policy

For each estimation of the advisor, the input data also varied in the same way that was described in previous evaluation (Sect. 4.1), which means that each policy has been executed 28,000 times for each inaccurate profile. After the execution, the advisor provided either the same decision that was computed using the accurate profile or a different one. Even if the decision was the same, it is usually based on slightly different results. Thus, we measured whether the decision is the same or not, and the relative difference between results using an inaccurate and the accurate profile. These relative differences, when executing the deadline-aware policy, were calculated as follows:

$$\frac{Cost_{Inaccurate} - Cost_{Accurate}}{Cost_{Accurate}} \tag{5}$$

where $Cost_{Inaccurate}$ and $Cost_{Accurate}$ are the costs measured using inaccurate and accurate profiles, respectively. The relative differences were calculated similarly when executing the budget-aware policy. Table 1 shows a comparison of the results provided by the advisor using inaccurate and accurate profiles. The first column compares the decision of the advisor using both profiles: = means same decision and \neq otherwise. For each policy, this table shows the average of the relative differences (avg), their standard deviation (std) and total number of decisions (size) for each inaccurate profile. So, for each inaccurate profile, the sum of equal and different decisions is 28,000.

As the error gets close to zero, the percentage of same decisions increases. Even when the error is 0.9, the advisor computed the same decision for deadline-aware policy around 93 % of the times and for budget-aware policy around 92 % of times. The number of different decisions for the budget-aware is greater than the number of same decisions (53 % against 47 %, respectively) only when the

Table 1. Results from the advisor using inaccurate and accurate profiles

Decision	Error	Deadline-aware			Budget-aware		
		Avg	Std	Size	Avg	Std	Size
=	-0.9	-0.8	0.3	17293 (62 %)	-0.2	0.7	13068 (47 %)
\neq		-1.0	0.0	10707 (38 %)	-0.4	0.5	14932 (53 %)
=	-0.5	-0.4	0.3	25356 (91 %)	0.3	0.6	23236 (83 %)
\neq		-0.6	0.0	2644 (9 %)	0.0	0.4	4764 (17 %)
=	-0.1	-0.1	0.1	26004 (93 %)	0.1	0.4	27272 (96 %)
\neq		0.1	0.3	1996 (7 %)	0.2	0.3	728 (4 %)
=	0.1	0.1	0.1	27115 (97 %)	0.1	0.1	27829 (99 %)
\neq		0.1	0.1	885 (3 %)	0.1	0.0	171 (1 %)
=	0.5	0.4	0.4	26597 (95 %)	0.1	0.3	26685 (95 %)
\neq		0.8	0.2	1403 (5 %)	0.2	0.2	1315 (5 %)
=	0.9	0.9	0.8	25982 (93 %)	0.0	0.3	25807 (92 %)
\neq		1.5	0.4	2018 (7 %)	0.3	0.3	2193 (8 %)

error is -0.9 . For this error, the advisor proposed the same decision for the deadline-aware policy around 62% of the time.

For most of the inaccuracy ranges, the number of equal decisions is far greater than the number of different decisions. That is, even if the application profile is inaccurate, it has a minor decision impact. Therefore, the advisor provides evidence that it is resilient to inaccuracies in the application profile. One reason for this is the wide spectrum of data that has been exercised. In some situations, the difference between choosing cloud or local is so great that the inaccuracy has little to no impact on job(s) placement decision. These results also show that as inaccuracy gets close to zero, the number of correct decisions increases, as expected. When the inaccuracy is close to zero, the infrastructure calculated by the application profile has low chance of being relevant to the final result.

5 Conclusions

The advisory service is composed of modules that can be extended/plugged-in to have more refined *Application Profiles* and to suit other applications. Further investigations will be required to collect data from a wide range of applications. The main lessons from our study are: (i) in HPC cloud, apart from resource performance, it is important to consider the time a user has to wait in a job queue of the on-premise environment compared to the overhead of cloud resources—more relevant is the total turnaround time, as also pointed by Marathe *et al.* [14]; (ii) it is possible to consider an advisory service for HPC hybrid clouds even without having highly precise application/job profiles—however, very inaccurate profiles may generate negative impact on costs and turnaround delays; (iii) the higher the cost differences between cloud and on-premise resources the higher the savings brought by an advisory service for resource selection on hybrid clouds.

Acknowledgment. We thank Eduardo Rodrigues and Nicole Sultanum for their comments on this paper. This work has been partially supported by FINEP/MCTI under grant no. 03.14.0062.00.

References

1. Belgacem, M.B., Chopard, B.: A hybrid HPC/cloud distributed infrastructure: coupling EC2 cloud resources with HPC clusters to run large tightly coupled multiscale applications. *Future Gener. Comput. Syst.* **42**, 11–21 (2015)
2. Binotto, A.P.D., Wehrmeister, M.A., Kuijper, A., Pereira, C.E.: Sm@rtConfig: a context-aware runtime and tuning system using an aspect-oriented approach for data intensive engineering applications. *Control Eng. Prac.* **21**(2), 204–217 (2013)
3. Calheiros, R.N., Netto, M.A.S., Rose, C.A.F.D., Buyya, R.: EMUSIM: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications. *Prac. Experience, Softw.* **43**(5), 595–612 (2013)

4. De Assunção, M.D., Di Costanzo, A., Buyya, R.: Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In: Proceedings of the ACM International Symposium on High Performance Distributed Computing (2009)
5. Delimitrou, C., Kozyrakis, C.: QoS-aware scheduling in heterogeneous datacenters with paragon. *ACM Trans. Comput. Syst.* **31**(4), 12 (2013)
6. Gentzsch, W., Yenier, B.: The UberCloud HPC experiment: compendium of case studies. Technical report, Tabor Communications, Inc. (2013)
7. Gentzsch, W., Yenier, B.: The UberCloud experiment: technical computing in the cloud - 2nd compendium of case studies. Technical report, Tabor Communications, Inc. (2014)
8. Gupta, A., Kale, L.V., Gioachin, F., March, V., Suen, C.H., Lee, B.S., Faraboschi, P., Kaufmann, R., Milojevic, D.: The who, what, why and how of high performance computing applications in the cloud. In: Proceedings of the IEEE International Conference on Cloud Computing Technology and Science (2013)
9. Gupta, A., Milojevic, D.: Evaluation of HPC applications on cloud. In: Open Cirrus Summit (2011)
10. Jarvis, S.A., Spooner, D.P., Keung, H.N.L.C., Cao, J., Saini, S., Nudd, G.R.: Performance prediction and its use in parallel and distributed computing systems. *Future Gener. Comput. Syst.* **22**(7), 745–754 (2006)
11. Li, H.: Workload dynamics on clusters and grids. *J. Supercomput.* **47**(1), 1–20 (2009)
12. Lowen, S.B., Teich, M.C.: *Fractal-Based Point Processes*. Wiley, New York (2005)
13. Mantripragada, K., Binotto, A., Tizzei, L.P.: A self-adaptive auto-scaling method for scientific applications on HPC environments and clouds. In: Proceedings of the International Workshop on Adaptive Self-tuning Computing Systems (2015)
14. Marathe, A., Harris, R., Lowenthal, D.K., de Supinski, B.R., Rountree, B., Schulz, M., Yuan, X.: A comparative study of high-performance computing on the cloud. In: Proceedings of the International Symposium on High-performance Parallel and Distributed Computing (2013)
15. Mateescu, G., Gentzsch, W., Ribbens, C.J.: Hybrid computing-where HPC meets grid and cloud computing. *Future Gener. Comput. Syst.* **27**(5), 440–453 (2011)
16. Napper, J., Bientinesi, P.: Can cloud computing reach the top500? In: Proceedings of the Combined Workshops on UnConventional High Performance Computing Workshop Plus Memory Access Workshop (2009)
17. Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T., Epema, D.: A performance analysis of EC2 cloud computing services for scientific computing. In: Avresky, D.R., Diaz, M., Bode, A., Ciciani, B., Dekel, E. (eds.) *Proceedings of Cloud Computing. LNICST*, vol. 34, pp. 115–131. Springer, Heidelberg (2010)
18. Sadjadi, S.M., Shimizu, S., Figueroa, J., Rangaswami, R., Delgado, J., Duran, H., Collazo-Mojica, X.J.: A modeling approach for estimating execution time of long-running scientific applications. In: Proceedings of the IEEE International Symposium on Parallel and Distributed Processing (2008)
19. Unuvar, M., Steinder, M., Tantawi, A.N.: Hybrid cloud placement algorithm. In: Proceedings of the IEEE International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (2014)
20. Vecchiola, C., Pandey, S., Buyya, R.: High-performance cloud computing: a view of scientific applications. In: Proceedings of the International Symposium on Pervasive Systems, Algorithms, and Networks (2009)
21. Virieux, J., Operto, S.: An overview of full-waveform inversion in exploration geophysics. *Geophysics* **74**(6), WCC1–WCC6 (2009)

22. Yang, L.T., Ma, X., Mueller, F.: Cross-platform performance prediction of parallel applications using partial execution. In: ACM/IEEE Supercomputing (2005)
23. Zaspel, P., Griebel, M.: Massively parallel fluid simulations on amazon's HPC cloud. In: Proceedings of the International Symposium on Network Cloud Computing and Applications (2011)
24. Zheng, G., Wilmarth, T., Jagadishprasad, P., Kalé, L.V.: Simulation-based performance prediction for large parallel machines. *Int. J. Parallel Program.* **33**(2), 183–207 (2005)

Optimizing Long-term IaaS Service Composition

Sajib Mistry, Athman Bouguettaya, Hai Dong^(✉), and A.K. Qin

School of Computer Science and Information Technology,
RMIT University, Melbourne, Australia
{sajib.mistry,athman.bouguettaya,hai.dong,kai.qin}@rmit.edu.au

Abstract. We propose a new economic model based optimization approach to compose an optimal set of infrastructure service requests over a long-term period. The service requests have the features of variable arrival time and dynamic resource and QoS requirements. A new economic model is proposed that incorporates dynamic pricing and operation cost modeling of the service requests. A genetic optimization approach is incorporated in the economic model that generates dynamic global solutions considering the runtime behavior of service requests. Experimental results prove the feasibility of the proposed approach.

1 Introduction

Cloud computing is increasingly becoming the technology of choice as the next-generation platform for conducting business. Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) solutions have already been offered by big companies in the cloud market [1]. An IaaS provider receives service requests in the form of computing resource (i.e., functional) and Quality of Service (QoS) (i.e., non-functional) requirements. The provider associates resource specifications (e.g., CPU, Memory, and Network Bandwidth) and QoS attributes (e.g., availability, throughput, and response time) with provided Virtual Machines (VMs) [2]. For the latter, the provider usually specifies them in a Service Level Agreements (SLA), where SLA violations may incur penalties for the provider [1]. The key challenge for the provider is the optimal composition of the service requests that closely meet the provider's economic expectation by considering certain constraints, such as resource limits and SLA violations.

The provider-consumer relationship between IaaS and SaaS providers is long-lasting and economically driven [3]. The contract period of a long-term service are usually counted in months or years. Due to the nature of the cloud consumers, e.g., multi-tenancy and changing business and cost requirements, a fixed set of requirements are often inapplicable in a long-term period. The key characteristic of the long-term service requests is that its *functional* and *non-functional* requirements change from time to time [3]. The long-term economic expectation of an IaaS provider also can change over time. Here we consider profit, SLA violations, and resource utilization as the key components in the long-term economic expectation. For example, a provider may find that SLA violations in the summer period influence the reputation more than the other periods. In this

research, we assume that the long-term economic expectation of an IaaS provider has already been defined. The objective is to select an optimal set of long-term service requests that satisfy the provider's long-term economic expectations.

In our previous research, we propose a new model for predicting dynamic consumer request behavior in long-term IaaS service compositions [4]. We observe that SaaS providers' run-time service requests can be different from their initial requests. However, an effective economic model based optimization process using the predicted data is missing in the existing approaches. For example, the under-utilized resources for some service requests can be reallocated to fulfill other suitable service requests to maximize the profit. To the best of our knowledge, existing composition approaches only consider the composition of short-term service requests [5, 6]. We identify that two long-term factors, i.e., *economic model* and *dynamic optimization* are missing in the existing approaches. The long-term economic model evaluates a composition of requests using the future predicted economic factors, such as dynamic pricing and operation cost. The dynamic optimization process operates at different times of a composition interval to improve the efficiency of the global composition.

We propose a new long-term IaaS service composition framework that composes stochastic service requests dynamically. We represent the long-term service requests as multidimensional time series. Existing approaches consider the short-term economic model of the IaaS provider [7–9], which is inapplicable for a long-term period. *We propose an economic model for cost and revenue analysis considering the long-term aspects of IaaS provider.* The proposed economic model is constructed as a Dynamic Bayesian Network (DBN) [10]. We use a multidimensional time series matching approach to evaluate a composition's fitness to the given long-term economic expectation. Next, *we propose a genetic algorithm (GA) for long-term service composition using the economic model.* Although combinatorial optimization techniques such as Integer Linear Programming (ILP) and dynamic programming are the preferable choices for service composition [4], they cannot be used in IaaS service combination, as the economic model is non-linear in nature [9]. The proposed approach deals with the non-linear objective function by creating a population of feasible solutions.

2 Related Work

A prediction model for long-term dynamic behavior of the consumers is proposed in [4]. A Mixed Integer Linear Programming (MILP) based optimization process is used with short-term economic and *one-time* arrival model of requests in [4]. Short-term resource allocation algorithms for SaaS providers are proposed in [5] to minimize the infrastructure cost and SLA violations. A task scheduling algorithm is proposed in [6] that uses analytic hierarchy process to optimally allocate resources. Heuristic algorithms are proposed to determine whether a new request can be admitted without impacting accepted requests in [11]. We propose a dynamic IaaS service composition framework that considers stochastic arrival of the requests and long-term economic model of the provider, which is a new area in this field.

The long-term QoS economic model of a service consumer is proposed in [2]. The model states how QoS requirements of end users change over time. An economic model for self-tuned catching is proposed in [7]. An economic model of federated cloud is described in [9]. The model evaluates the cost of using resources from a federated cloud and develops a resource management core for the profit maximization. Existing economic models, nevertheless, do not consider long-term relationships among composite QoSs, resource utilization, and operation costs. Generic penalty-based and repair-based methods for composition are proposed in [12]. However, such methods are not incorporated in long-term economic models for dynamic environments. Hence, we need to develop an economic model based dynamic optimization technique for IaaS service composition.

3 The Long-term Economic Model of the IaaS Provider

We represent the long-term service requests as multidimensional time series. Each long-term service request of the i^{th} consumers is defined as a tuple, $U_i = \{c_t, m_t, nb_t, av_t, rt_t, th_t \mid t \in [1, T]\}$. Here, $c, m,$ and nb are the required units of CPU, Memory, and Network respectively. The QoS requirements $av, rt,$ and th specify the required units of Availability, Response time, and Throughput respectively. $[1, T]$ is a composition interval. Let us assume that there are k requests (S_k) in the map denoted as $MAP(S_k) = \{ \langle U_1, Arrival_1 \rangle, \dots, \langle U_k, Arrival_k \rangle \}$. The requests are already transformed into runtime behavior and the arrival times are predicted in a request map. The task of the economic model is to generate the long-term economic valuation of the request map. We consider profit (P), number of SLA violations (NO_SLV), and resource utilization factor (UF) as the key attributes in the economic valuation. The next task is to measure the closeness of the composition to a given long-term economic expectation.

3.1 The Long-term Economic Valuation

The first task is to convert the request map $MAP(S_k)$ to the long-term economic valuation denoted as $EVAL(t) = \{P_t, NO_SLV_t, UF_t \mid t \in T\}$. We use long-term revenue and operation cost modeling to calculate the profit. As the business models of IaaS providers are similar to business models of utility providers, *demand-driven pricing model* is a common phenomena in the cloud market. For example, the price of a EC2 service fluctuates up to 80 % in the Amazon cloud spot market [9]. As Dynamic Bayesian Network (DBN) models succeed in modelling temporal dynamic environments [10], we represent the dynamic pricing behavior as a DBN. The DBN describes the correlations among physical resources (computing (C), storage(M) and network(NB)), QoS values(Availability (AV), Throughput (TH), and Response time (RT)), demand, and service price. The model calculates the probability (Q) of a service price X at t for the service request $U^{(t)} = \{C, M, NB, AV, TH, RT\}$ given its previous price at time $(t - 1)$ as follows:

$$Q(X, U^{(t)}, t) = P(Price_t = X \mid U^{(t)}, Price_{t-1}) \quad (1)$$

The IaaS provider may serve exiting and new requests at the same time. Due to the pricing model, a service may be priced differently for different consumers. As long-term requests reserve the resources at current prices, the price is correlated with the arrival time. For instance, a consumer who arrives at time t always pays the price advertised at time t . As each request in $MAP(S_k)$ is associated with its arrival time, we sort them in an ascending order. Let us denote, $Q(X, Y, U^{(t)}, t_0)$ is the initial probability table for the request U_0 . We generate the individual probability table $Q(X, Y, U^{(t)}, t_i)$ for request U_i using Eq. 1. We calculate the revenue time series (REV_t) of $MAP(S_k)$ using the Maximum Probable Explanation (MPE) algorithm [10] on $Q(X, Y, U^{(t)}, t_i)$ in Eq. 2.

$$P_i \dots P_0 = \arg \max \{Q(X, U^{(t)}, t_i) \mid Q(X, U^{(t)}, t_{i-1}), \dots, Q(X, U^{(t)}, t_0)\} \tag{2}$$

$$REV_t = \sum_i^{|MAP(S_k)|} P_i, \text{ where } U^{(t)} \in U_i$$

We apply the following assumptions to model the long-term cost:

1. **The fixed costs of the provider are distributed over the servers for the amortization period.** Such costs include data center building cost, land cost, hardware price, etc. Each server i has a capital cost in a time unit t represented as $CAPEX_{it}$. Generally t is treated as a month or quarter in a year. If the provider has N physical servers, the fixed cost at time t is calculated by the following equation:

$$Fixed_Cost_t = CAPEX_{it} \times N. \tag{3}$$

2. **The operation cost is determined by the server power consumption.** Although there are other costs, such as maintenance cost, employee salary, and insurance cost, we choose the power consumption cost for simplicity. ARIMA is a popular model to predict the dynamic price of power [9]. In Eq. 4, the auto regressive (AR) part depends on the p lagged values of the time series of aggregated AG_n where L is the lag operator and α_i is the coefficient constant. The moving average (MA) part depends on the q lagged values of the previous prediction errors (ϵ_t) and θ_i is the coefficient constant. d represents the number of times that the difference operation is performed to obtain the stationary time series. The values of (p, d, q) is determined by the Box-Jenkins method for the price history [4].

$$(1 - \sum_{i=1}^p \alpha_i L^i)(1 - L)^d Power_t = (1 + \sum_{i=1}^q \theta_i L^i) \epsilon_t. \tag{4}$$

3. **Each running physical server should have a predefined threshold (σ) of utility.** For example, the provider may decide that physical server will not run unless the resources are expected to be used more than 30 %.

The number of physical servers that will run to satisfy requests depends on the resource allocation module installed in the server [9]. The module requires the composed request from individual service requests. The composed request time series ($\{\hat{C}_t, \hat{M}_t, \hat{N}B_t, \hat{A}V_t, \hat{T}H_t, \hat{R}T_t\}$) for $MAP(S_k)$ can be formed using the composition rule of resources [5] and composition rules of the QoS [3] as follows:

$$\text{Resource Composition: } \bar{X}_t = \sum_i^{|\hat{M}\hat{A}P(S_k)|} X_i^{(t)} \text{ where, } X = \{C, M, NB\} \quad (5)$$

$$\text{QoS Composition: } \bar{A}V_t = \max(av_i^{(t)}); \bar{T}H_t = \max(th_i^{(t)}); \bar{R}T_t = \sum_i^{|\hat{M}\hat{A}P(S_k)|} (rt_i^{(t)}); \text{ where } i \in \hat{M}\hat{A}P(S_k)$$

Our long-term economic model is independent of any particular resource allocation schemes. We assume that the resource allocation scheme installed in the physical servers has a function (F) to convert the composed requests to an utility factor (UF) as:

$$UF_t = F(\bar{C}_t, \bar{M}_t, \bar{N}B_t, \bar{A}V_t, \bar{T}H_t, \bar{R}T_t) \quad (6)$$

Let us assume that a server has a maximum utility factor UF_{max} . Hence, $\lceil \frac{UF_t}{UF_{max}} \rceil$ of the physical server is needed to satisfy the composite requests. Each running physical server has two types of power cost units: (a) fixed power cost for the routine operation of the server (*fixed_unit*) and (b) variable power cost (*variable_unit*) to satisfy the utility factor. If UF is linearly proportional to the (*variable_unit*) with the constant Var_{cons} , we can calculate the operation cost at time t using the following equation:

$$OP_Cost_t = Power_t \times (fixed_unit \times \lceil \frac{UF_t}{UF_{max}} \rceil + Var_{cons} \times UF_t) \quad (7)$$

The proposed framework allows SLA violations to occur. The SLA violation cost $SLA_Cost(t)$ depends on the SLA violation penalty rate ($SLA_Penalty$) and the number of SLA violations (NO_SLV_t) in a certain time t . We calculate the number of SLA violations and the constrained satisfied utility factor using Algorithm 1. Algorithm 1 checks whether a composition satisfies all the constraints. It reduces the number of service requests until all the constraints are satisfied. The SLA violation cost (SLA_Cost) is calculated using the following equation:

$$SLA_Cost_t = SLA_Penalty \times NO_SLV_t \quad (8)$$

The revenue, the operation cost, and the SLA violation cost are used to calculate the profit of a composition. Algorithm 1 determines the number of SLA violation and the resource utilization factor. We generate long-term economic valuation of the request map $MAP(S_k)$ using Eqs. 2, 3, 7, and 8 as follow:

$$EVAL(t) = \{REV_t - Fixed_Cost_t - OP_Cost_t - SLA_Cost_t, NO_SLV_t, UF_t \mid t \in T\}. \quad (9)$$

3.2 Long-term Economic Expectation and Fitness of a Composition

We include profit, number of SLA violation, and resource utility factor as the key attributes in the long-term economic expectation. However, the influence of the attributes may not be equal all the time. For example, SLA violations at peak hours may be more important than the profit when considering business reputation. We incorporate such influence weights in the multidimensional time series, $EXP(t) = \{(P_t^E, W_t^P), (NO_SLV_t^E, W_t^S), (UF_t^E, W_t^{UF}) \mid t \in T\}$. We define the fitness of a composition as the distance between the long-term economic valuation of

Algorithm 1. Determining the number of SLA violation and Utility Factor of a composition

Input: Request Map: $MAP(S_k)$, Resource and QoS constraints: $(C_{max}, M_{max}, NB_{max}, AV_{max}, RT_{max}, TH_{max})$, server utility threshold σ and maximum utility factor UF_{max} .
Output: the number of SLA violations (NO_SLV_t) and Utility Factor (UF_t) at time t

- 1: $NO_SLV_t := -1$
- 2: **repeat**
- 3: $NO_SLV_t := NO_SLV_t + 1$
- 4: Set $Candidate(|MAP(S_k)|) = \{i \in MAP(S_k)\}$
- 5: Generate $X_t := (C_t, M_t, NB_t, AV_t, TH_t, RT_t)$ using $Candidate(|MAP(S_k)|)$ in Eq. 5
- 6: Generate $UF_t := F(Y_t)$ using Eq. 6
- 7: Server utility threshold $\hat{\sigma} := (UF_{max} \bmod UF_t)$
- 8: Remove any requests from the $MAP(S_k)$ with a random distribution and update $|MAP(S_k)| := |MAP(S_k)| - 1$
- 9: **until** $X_t < X_{max} \mid X \in \{C, M, NB, AV, RT, TH\}$ and $\hat{\sigma} < \sigma$
- 10: Return NO_SLV_t and UF_t

the composition and the economic expectation. For simplicity, we deploy the Euclidean Distance as the default distance function. As smaller distance infers better fitness, we formulate the weighted fitness function in Eq. 10. This fitness function acts as the objective function in the proposed genetic optimization.

$$Fitness(MAP(S_k)) = DIS(EXP, EVAL) = \sqrt{\sum_{t=1}^T \frac{W_t^P}{(P_t^E - P_t)^2} + \frac{W_t^S}{(NO_SLV_t^E - NO_SLV_t)^2} + \frac{W_t^{UF}}{(UF_t^E - UF_t)^2}}. \quad (10)$$

3.3 Genetic Optimization Using the Economic Model

The proposed economic model is non-linear in nature. We design a genetic algorithm (GA) to address the non-linear properties in IaaS economic model. The task of the optimization process is to find an optimal subset $\{MAP(S_k) \mid k < N\}$ that maximizes the fitness of the composition stated in Eq. 10. The GA simulates evolutionary processes by considering an initial *population of individuals* and applying genetic operators in each reproduction. In optimization terms, each individual in the population is encoded into a string or *chromosome* that represents a possible solution to a given problem. We use a binary string representation $X[j] = 0$ or 1 to represent the possible solution of $MAP(S_k)$. The j^{th} request is considered in the solution if $X[j] = 1$. For example, a candidate solution $\{B, C\}$ is represented as $\{0110\}$ in the IaaS composition if the incoming requests are $\{A, B, C, D\}$. The fitness of an individual is evaluated based on the fitness function in Eq. 10. Highly fit solutions reproduce new “offspring” solutions (i.e., children) by exchanging pieces of their genetic information in a *crossover* procedure. Mutation is often applied after crossover by altering some genes in the strings. The less fit chromosomes in the population are replaced by the new children. This process is repeated until a satisfactory solution is found. We use a binary tournament selection method [12] to generate the initial population for the first optimization. Crossover point is set in the middle point of chromosome. We set the mutation rate as 2 bits per child. After discarding the duplicated child in crossover operation, the new population replaces the individual chromosomes with the lowest fitness value (steady-state replacement). We continue generating new populations until the solutions are not improved.

The requests in the long-term IaaS composition are based on their predicted future behavior. However, existing requests in the system may behave differently than predicted. We use fixed interval (ΔT) to check whether the existing composition is deviating from the prediction. Other check points are the predicted incoming arrival times of new requests. For example, assume that $\{(A, t_0), (C, t_5)\}$ is the initial IaaS composition. If $\Delta t = 2$, the optimization check points are t_3 and t_5 . If request B arrives at time t_2 , we may consider inclusion of B if the new predicted behavior of A is different from its initial prediction. Hence, the optimization process should be run again from scratch to update the solution.

4 Experiments and Results

A set of experiments are conducted to evaluate the efficiency of the proposed long-term economic model and genetic algorithm. At first we evaluate the prediction accuracy of the economic model. The fitness of the economic model based composition is compared with a greedy and brute-force approach. In the greedy approach, the provider does not consider the future arrival and valuation of the requests. It accepts the requests that does not violate the SLA at the time of arrival [11]. The brute-force approach considers all the possible compositions to choose the optimal composition. All the experiments are conducted on computers with Intel Core i7 CPU (2.13 GHz and 4 GB RAM). R statistical tool [13] is used to implement the algorithms. We evaluate the proposed method using a mixture of Google Cluster resource utilization [14], real world cloud QoS performance [15], and synthetic data. We randomly generates 100 arrival time points and attach them with the requests to generate a stochastic map of 100 long-term requests. We synthetically generate the runtime behavior of the service requests using Correlation Density Index (CDI) [4]. A higher CDI refers to a lower randomness of the correlations between the service request and the runtime service usage in the history. We generate five sets of service runtime requests with 5 different CDI values (0.5, 0.6, 0.7, 0.8, 0.9) from the map of 100 long-term requests. The long-term economic expectation is also synthetically created (Fig. 1(a)). The mean and standard deviation of the attributes in economic expectation are as follows: profit (mean: \$100, sd: \$20, weight: 0.5), number of SLA violation (mean: \$3, sd: \$2, and weight: 0.3), and utility factor (mean: 80 %, sd: 30 %, and weight: 0.2).

4.1 Setup of the Long-term Economic Model

The proposed economic model incorporates the change of service price and operation cost. We use the time series of electricity price (E_t) from Australian Bureau of Statistics [16] to generate the change in the service price and operation cost. The price of resources and QoS are set by following a weighted Rackspace pricing model as ($\$5 \times E_t$ /unit per hour for any types of resources). [9] provides a short-term mapping relationship between operation cost and utilization. We

Table 1. Long-term relationship between Utility Factor and Operation Cost

UF	Cost/hour	UF	Cost/hour	UF	Cost/hour
5 %	$\$110 \times E_t$	20 %	$\$120 \times E_t$	30 %	$\$140 \times E_t$
60 %	$\$150 \times E_t$	90 %	$\$160 \times E_t$	100 %	$\$165 \times E_t$

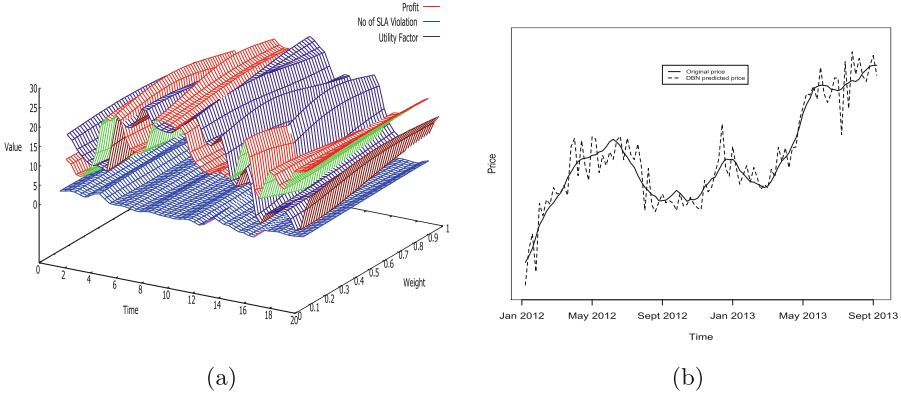


Fig. 1. (a) Long-term Economic Expectation of the provider (b) Prediction accuracy of DBN

generate the long-term behavior between operation cost and utilization using the dynamic electricity price in Table 1. The SLA violation fee is 20 % of the revenue credited to consumers. The resource constraints are set by allowing maximum 100 units for each attributes. We use the resource allocation algorithm in [11] as the utility factor in Eq. 6.

4.2 Efficiency of the Economic Model Based Composition

At first, we evaluate the performance of the propose DBN in the economic model. We model the 18 months (Jan 2012–Sep 2013) electricity price with the proposed DBN. The predicted price from the DBN is compared with the actual price using Normalized Root Mean Square Error (NRMSE) defined in Eq. 11.

$$NRMSE = \sqrt{\frac{\sum_{i=1}^n (\frac{PR(q) - AC(q)}{AC(q)_{max} - AC(q)_{min}})^2}{n}} \tag{11}$$

Figure 1(b) depicts the closeness (0.3 NRMSE) of the prediction toward actual behavior. Figure 2(a) depicts the efficiency of the long-term composition over the greedy approach. We observe that the greedy approach only maximizes the economic fitness at the beginning, while the proposed economic model based service composition maximizes the fitness in a steady rate over the period of composition (Fig. 2(a)). Although brute-force produces the best composition in Fig. 2(b), it is not applicable as its time complexity is (2^N) . The proposed approach is not polynomial with respect to request size (Fig. 2(a)). Although it takes

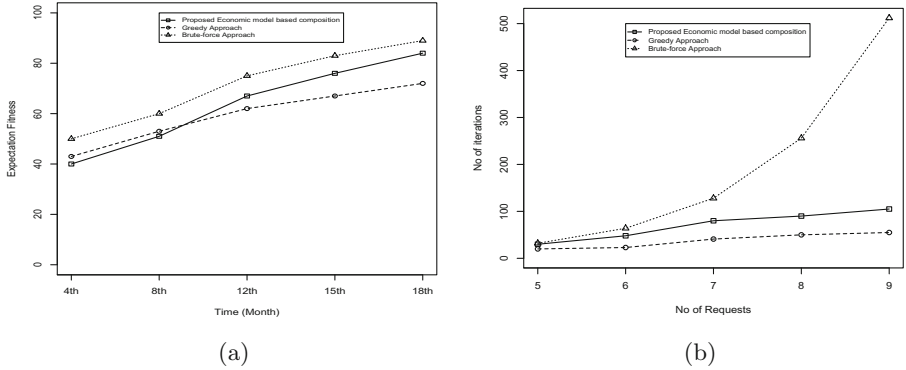


Fig. 2. (a) The long-term composition fitness (b) Effect of request size in time complexity

more computation time than the greedy approach, the difference is negligible in real world implementation.

5 Conclusion

In summary, we propose a novel dynamic service composition framework for IaaS providers to gain their long-term economic expectations. The proposed long-term economic model evaluates the long-term economic behavior of service compositions. Experimental results show that our approach is more profitable than the greedy approach and suitable for runtime implementation.

Acknowledgements. This research was made possible by NPRP 7-481-1-088 grant from the Qatar National Research Fund (a member of The Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A.: Above the clouds: A Berkeley view of cloud computing. Tech, UC Berkeley (2009)
2. Ye, Z., Bouguettaya, A., Zhou, X.: QoS-aware cloud service composition based on economic models. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) Service Oriented Computing. LNCS, vol. 7636, pp. 111–126. Springer, Heidelberg (2012)
3. Ye, Z., Mistry, S., Bouguettaya, A., Dong, H.: Long-term qos-aware cloud service composition using multivariate time series analysis. *IEEE Trans. Serv. Comput.* 1(99), 1–16 (2014)
4. Mistry, S., Bouguettaya, A., Dong, H., Qin, A.K.: Predicting dynamic requests behavior in long-term iaas service composition. In: Proceedings of ICWS (2015)
5. Wu, L., Garg, S., Buyya, R.: Sla-based resource allocation for software as a service provider (saas) in cloud environments. In: Proceedings of CCGrid, pp. 195–204 (2011)

6. Ergu, D., Kou, G., et al.: The analytic hierarchy process: task scheduling and resource allocation in cloud. *J. Supercomput.* **64**, 835–848 (2013)
7. Dash, D., Kantere, V., Ailamaki, A.: An economic model for self-tuned cloud caching. In: *Proceedings of ICDE*, pp. 1687–1693 (2009)
8. Thanakornworakij, T., Nassar, R. et al.: An economic model for maximizing profit of a cloud service provider. In: *Proceedings of ARES*, pp. 274–279 (2012)
9. Goiri, Í., Guitart, J., Torres, J.: Economic model of a cloud provider operating in a federated cloud. *Inf. Syst. Front.* **14**(4), 827–843 (2012)
10. Murphy, K.P.: “Dynamic bayesian networks: representation, inference and learning,” Ph.D. dissertation, University of California, Berkeley (2002)
11. Wu, L., Garg, S., Buyya, R.: Sla-based admission control for a software-as-a-service provider in cloud computing environments. *J. Comput. Syst. Sci.* **78**(5), 1280–1299 (2012)
12. Ye, Z., Zhou, X., Bouguettaya, A.: Genetic algorithm based QoS-aware service compositions in cloud computing. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) *DASFAA 2011, Part II. LNCS*, vol. 6588, pp. 321–334. Springer, Heidelberg (2011)
13. Qian, H.: PivotalR: a package for machine learning. *R J.* **6**(1), 57 (2014)
14. Reiss, C., Wilkes, J., Hellerstein, J.L.: Google cluster-usage traces: format + schema. Google Inc., Mountain View, CA, USA, Technical report (2011)
15. Jiang, W., Lee, D., Hu, S.: Large-scale longitudinal analysis of soap-based and restful web services. In: *Proceedings of ICWS*, pp. 218–225 (2012)
16. ESAA, “Electricity prices in australia,” *Aus B. of Statistics*, Technical report 02 (2000)

QoS and Trust (Short Papers)

On the Complexity of QoS-Aware Service Selection Problem

Faisal N. Abu-Khzam¹, Cristina Bazgan^{2,3}, Joyce El Haddad²,
and Florian Sikora²(✉)

¹ Departement of Computer Science and Mathematics,
Lebanese American University, Beirut, Lebanon
faisal.abukhzam@lau.edu.lb

² PSL, Université Paris-Dauphine, LAMSADE UMR CNRS 7243, Paris, France
{bazgan,elhaddad,florian.sikora}@lamsade.dauphine.fr

³ Institut Universitaire de France, Paris, France

Abstract. This paper addresses the QoS-aware service selection problem considering complex workflow patterns. More specifically, it focuses on the complexity issues of the problem. The NP-hardness of the problem, under various settings, has been open for many years and has never been addressed thoroughly. We study the problem complexity depending on the workflow structure, the number of workflow tasks, the number of alternative services per task and the categories of quality of service criterion associated to services. We provide for the first time the NP-hardness proof of the problem. Additionally, we show that the problem is polynomial in case of only one criterion per task and pseudo-polynomial if there is a fixed number of criteria.

Keywords: Quality of Service · Service selection · Optimization · Complex workflows

1 Introduction

One of the key features of Service-Oriented Architectures is to realize workflows by composing loosely coupled web services. Each of these services provides a functionality and their composition creates the functionality of a new value-added service, called a composite service. As many providers might offer functionally equivalent services, several candidates might be available for the realization of a task in a workflow. To distinguish among these candidates, their non-functional properties are considered such as Quality of Service (QoS) (e.g. execution duration, execution cost, reputation, and availability). Since only one component service is needed for a task, the service selection problem based on quality of service, denoted QOS-AWARE SERVICE SELECTION problem, is an important step in the composition process. It helps to choose services that best meet QoS constraints.

In the literature, the QOS-AWARE SERVICE SELECTION problem has been claimed to be NP-hard and then essentially exponential time exact algorithms

and heuristics have been investigated. Our first contribution in this paper is to establish that QOS-AWARE SERVICE SELECTION is solvable in polynomial time for one criterion for complex workflow including those with inclusive patterns (OR pattern) which have never been studied as far as we know. Another contribution is to present a first NP-hardness proof of the problem for workflows with simple structure even with two criteria. Moreover, our approach goes beyond classical QoS categories and adds new ones. However, we show that this problem is only *weakly* NP-hard, which allows the existence of an exact pseudo-polynomial time algorithm. This type of algorithm is of particular interest since typical values for real instances are small and in this case a pseudo-polynomial time algorithm becomes a polynomial time algorithm. Therefore, the problem is not as hard as claimed by previous studies. Our results are thus tight in the sense that our pseudo-polynomial time results are the best possible considering the complexity lower bounds.

Related Work. QOS-AWARE SERVICE SELECTION problem has been formulated as an optimization problem and discussed by several authors [1–3, 9, 10, 12, 14, 15]. Surveys of existing approaches can be found in [5, 8, 11]. Most of these papers note that the problem is NP-hard. They state that QOS-AWARE SERVICE SELECTION is equivalent to MULTI-DIMENSION MULTIPLE-CHOICE KNAPSACK problem [2]. To solve QOS-AWARE SERVICE SELECTION problem, some approaches propose exact solutions. Bonatti and Festa [3] formulated the problem as a matching problem between requests and offers. The goal is to find a binding between requests and offers, compatible with the given matching and optimal regarding the preferences associated to services and invocations. Yu and Lin [14] proposed a combinatorial approach modeling the problem as a MULTIPLE-CHOICE KNAPSACK problem applied only to sequential structure workflows, and a graph approach modeling the problem as a CONSTRAINED SHORTEST PATH problem applied to more general structure workflows. Schuller et al. [9] formulated the problem as a linear optimization problem for simple structure workflows, which can be solved optimally using integer linear programming techniques. This approach was extended in Schuller et al. [10] to consider structured as well as unstructured workflows. More recently, Gabrel et al. [6] studied complexity of the problem in case of one criteria and proposed a mixed integer program to solve it. Likewise, a number of heuristic algorithms have been proposed to solve QOS-AWARE SERVICE SELECTION. Zeng et al. [15] proposed a local optimization approach which selects Web services one at the time by associating a task of the workflow to the best candidate service. Canfora et al. [4] proposed to tackle the problem with a genetic algorithm. Zhang et al. [16] proposed a strategy to decompose a composite service with a general flow structure into parallel execution paths. Then, the authors modeled service selection problem for each execution path as a multi-objective optimization problem, and presented an ant colony optimization algorithm to solve it. More recently, Trummer et al. [12] proposed three algorithms to solve the problem: a first exact algorithm with exponential complexity, a second polynomial time heuristic algorithm with no guaranteed error bound, and a third polynomial time approximation algorithm with guarantee error bound.

Organization. The rest of the paper is organized as follows. Section 2 presents some background elements related to our problem, as well as definitions. Section 3 provides some positive complexity results. These positive results are tight as Sect. 4 shows complexity lower bounds. Section 5 concludes and highlights some future work. Due to space constraints, proofs are devoted to the full version of the paper.

2 System Model and Problem Statement

2.1 Workflows

A workflow is an abstract business process that combines several tasks. A task represents a step that has to be accomplished by a single web service invocation and is associated with a set of service candidates that are all able to provide the required functionality but might differ in their non-functional properties, namely Quality of Service (QoS) values. The connections (or transitions) between tasks represent a transfer of control from a preceding task to the one that follows. These connections include control structures such as sequence, parallel (AND-split, AND-join), exclusive (XOR-split and XOR-join), and inclusive (OR-split and OR-join) patterns (for details see in [13]). These patterns, written in Business Process Modeling Notation 2.0 could be concatenated or interlaced recursively to create complex workflows.

In this work, we expect all workflows to be *structured* [7] that is meeting the following requirements : (i) having a single *entry* point (i.e. with no incoming connection), (ii) a single *exit* point (i.e. with no outgoing connections), and (iii) a split of some kind is *closed* by a corresponding join of the same kind and the same number of branches (i.e. each AND-split has a corresponding AND-join and each (X)OR-split has a corresponding (X)OR-join). Formally, a structured workflow is defined as follows.

Definition 1 (Structured Workflow). *A structured workflow wf on a set of tasks $T = \{t_1, \dots, t_n\}$ is defined recursively as:*

- a task $t_i \in T$ is a structured workflow wf . Both the entry and exit points of wf are t_i in this case.
- a sequence pattern (wf_1, \dots, wf_ℓ) of structured workflows wf_1, \dots, wf_ℓ . The entry point of wf is the entry point of wf_1 , the exit point of wf is the exit point of wf_ℓ , and for all $k, 1 \leq k < \ell$, the exit point of wf_k has a transition to the entry point of wf_{k+1} .
- an AND pattern of structured workflows AND-split(wf_1, \dots, wf_ℓ)AND-join where the entry point of wf is AND-split, the exit point of wf is AND-join, and with transitions between AND-split and the entry points of each wf_i and between the exit point of each wf_i and AND-join.
- a XOR pattern of structured workflows XOR-split(wf_1, \dots, wf_ℓ)XOR-join where the entry point of wf is XOR-split, the exit point of wf is XOR-join, and with transitions between XOR-split and the entry points of each wf_i and between the exit point of each wf_i and XOR-join.

- an OR pattern of structured workflows $OR\text{-}split(wf_1, \dots, wf_\ell)OR\text{-}join$ where the entry point of wf is $OR\text{-}split$, the exit point of wf is $OR\text{-}join$, and with transitions between $OR\text{-}split$ and the entry points of each wf_i and between the exit point of each wf_i and $OR\text{-}join$.

Definition 2 (Structured Workflow Without Sharing Tasks). *A structured workflow without sharing is a structured workflow in which every task has one incoming and one outgoing transition.*

We represent workflows as trees of patterns recursively as in Definition 1 where a leaf node represents a task and a non-leaf node is either SEQ for sequential pattern, AND for parallel pattern, XOR (resp. OR) for a choice of one (resp. of one or several) out of several alternative branches.

The control flow describes the execution ordering of the tasks through different patterns. Informally, a subset of tasks $T' \subseteq T$ satisfies the control flow of a workflow wf if T' satisfies the control flow of at least one wf_i in the case of OR pattern, of all wf_i in the case of AND or sequence patterns, or of exactly one wf_i in the case of XOR pattern. More formally:

Definition 3 (Control Flow Satisfaction). *Consider a workflow wf on a set of tasks T . The control flow satisfaction of wf by a subset of tasks $T' \subseteq T$ is defined recursively as:*

- when wf is a task t_i , then T' satisfies the control flow of wf iff $t_i \in T'$.
- when wf is a sequence of workflows (wf_1, \dots, wf_ℓ) , then T' satisfies the control flow of wf iff T' satisfies the control flow of each workflow wf_i .
- when wf is $AND\text{-}split(wf_1, \dots, wf_\ell)AND\text{-}join$, then T' satisfies the control flow of wf iff T' satisfies the control flow of each workflow wf_i .
- when wf is $XOR\text{-}split(wf_1, \dots, wf_\ell)XOR\text{-}join$, then T' satisfies the control flow of wf iff T' satisfies the control flow of exactly one workflow wf_i .
- when wf is $OR\text{-}split(wf_1, \dots, wf_\ell)OR\text{-}join$, then T' satisfies the control flow of wf iff T' satisfies the control flow of at least one workflow wf_i .

2.2 Services and Quality of Service

Services are software components that encapsulate atomic functionality. Since executing a task only one service is needed among service candidates then non-functional properties such as Quality of Service (QoS) are considered. Denoting by \mathcal{C} the set of criteria and assuming a fixed ordering between them, services are described by their QoS vectors of non-negative rational values as follows.

Definition 4 (Service). *A service s is a tuple (f, c) where f represents its offered functionality, and $c = (c_1(s), \dots, c_p(s))$ represents its QoS vector where $c_i(s)$ is the QoS value for criterion i .*

Consider a workflow wf composed of a set of n tasks $T = \{t_1, \dots, t_n\}$. For each task $t_j \in T$, there is a set of m_j candidate services $S_j = \{s_{j,1}, \dots, s_{j,m_j}\}$.

A vector $c(s_{j,\ell}) = (c_1(s_{j,\ell}), \dots, c_p(s_{j,\ell}))$ is associated to each service $s_{j,\ell}$ that represents the evaluation of a service $s_{j,\ell}$ on criterion c_i , for $i = 1, \dots, p$. Consider the set $T' \subseteq T$ satisfying the control flow of wf and the set S of selected services, we can compute the QoS values of a workflow wf from the QoS values of the selected services $s_{j,\ell} \in S$ as shown in Table 1. Note that, for an exclusive pattern (XOR-split/-join pattern) there is no aggregation function between wf_i , $i = 1, \dots, \ell$ since exactly one wf_i satisfies the control flow. For all the other patterns, the QoS of the wf is the QoS of the selected services for the workflows wf_i that satisfy the control flow. As shown in Table 1, we classify into five categories the most used QoS criteria according to their aggregation functions that could be summation (\sum or *sum*), product (\prod or *prod*), minimum (*min*) and maximum (*max*). Optimization objective could be minimization (*MIN*) or maximization (*MAX*).

Table 1. Aggregation Functions

QoS category	Sequence pattern	AND or OR pattern	Objective function
cat 1	$\sum_{t_j \in T', s_{j,\ell} \in S \cap S_j} c_i(s_{j,\ell})$	$\sum_{t_j \in T', s_{j,\ell} \in S \cap S_j} c_i(s_{j,\ell})$	MIN or MAX
cat 2	$\sum_{t_j \in T', s_{j,\ell} \in S \cap S_j} c_i(s_{j,\ell})$	$\max\{c_i(s_{j,\ell}) : t_j \in T', s_{j,\ell} \in S \cap S_j\}$	MIN
cat 3	$\prod_{t_j \in T', s_{j,\ell} \in S \cap S_j} c_i(s_{j,\ell})$	$\prod_{t_j \in T', s_{j,\ell} \in S \cap S_j} c_i(s_{j,\ell})$	MIN or MAX
cat 4	$\min\{c_i(s_{j,\ell}) : t_j \in T', s_{j,\ell} \in S \cap S_j\}$	$\min\{c_i(s_{j,\ell}) : t_j \in T', s_{j,\ell} \in S \cap S_j\}$	MAX
cat 5	$\max\{c_i(s_{j,\ell}) : t_j \in T', s_{j,\ell} \in S \cap S_j\}$	$\max\{c_i(s_{j,\ell}) : t_j \in T', s_{j,\ell} \in S \cap S_j\}$	MIN

The QoS values of a leaf in the tree of patterns are equal to the QoS values of the selected service $s_{j,\ell}$ in the candidate set S_j . The QoS values of a non-leaf node are computed based on the QoS criteria values of its children and their associated aggregation functions. The QoS values of non-leaf nodes are computed recursively.

2.3 Problem Statement

The QOS-AWARE SERVICE SELECTION problem is an optimization problem that aims at binding every leaf node in the tree of patterns representing a workflow task to at most one of its service candidates. Formally, the problem can be stated as follows.

The QoS range values of a criterion c_i of category *cat 3* is in $\mathbb{Q} \cap (0, 1]$, and of categories *cat 1*, *cat 2*, *cat 4* and *cat 5* is in \mathbb{N} for any service s .

The decision problem associated with QOS-AWARE SERVICE SELECTION consists of p bounds $b_1, \dots, b_p \in \mathbb{Q}$ to decide if there exists a subset of tasks $T' \subseteq T$ satisfying the control flow of wf and a service $s_{j,\ell} \in S_j$ associated to each task $t_j \in T'$ such that the value c_i associated with this solution is smaller than or

QOS-AWARE SERVICE SELECTION:

Input: A workflow wf on a set of tasks $T = \{t_1, \dots, t_n\}$, a set of candidate services S_j for each task $t_j \in T$ (of size at most m) and a vector $c = (c_1, \dots, c_p)$ of p criteria such that $c_i(s) \in \mathbb{Q}$, $\forall s \in S_j$, $j = 1, \dots, n$, $i = 1, \dots, p$.

Output: A subset of tasks $T' \subseteq T$ satisfying the control flow of wf and a subset S of services such that a service $s_{j,\ell} \in S_j \cap S$ is associated to each task $t_j \in T'$ in order to optimize each c_i , $1 \leq i \leq p$.

equal to (resp. greater than or equal to) b_i in the case of a minimization (resp. maximization) criterion c_i , for $i = 1, \dots, p$.

The solution for our problem is a subset $T' \subseteq T$ containing only leaves of the tree of patterns (i.e. only tasks) satisfying the control flow of wf in contrast with the other papers in the area where all leaves of the tree (i.e. all tasks of T) are selected in T' even if no service is selected for a task.

Before giving some complexity notions, we define the size on an input of QOS-AWARE SERVICE SELECTION. Given an input instance I of the problem, its maximum value, denoted by $V(I)$, is $\max\{c_i(s) : s \in S_j, j = 1, \dots, n, i = 1, \dots, p\}$ and its size, denoted by $|I|$, is $O(nm \log V(I))$.

3 Complexity Upper Bounds

In this section, we give some positive results concerning QOS-AWARE SERVICE SELECTION. We show that the problem is polynomial time solvable if there is only one criterion and pseudo-polynomial time solvable if there are a fixed number of criteria. Moreover, we will show in the next section that pseudo-polynomiality is also tight, i.e. one cannot expect polynomiality.

When the number of criteria is one (i.e. $p = 1$), then QOS-AWARE SERVICE SELECTION is a single criterion optimization problem and we search for an optimal solution. When the number of criteria is more than one ($p \geq 2$), then QOS-AWARE SERVICE SELECTION is a p -criteria optimization problem. In this case, there is typically no optimal solution that is the best for all the criteria. Therefore, the standard situation is that any solution can always be improved on at least one criterion. The solutions of interest, called efficient solutions, are those such that any other solution which is better on one criterion is necessarily worse on at least one other criterion. The vectors associated to efficient solutions are called nondominated vectors.

Theorem 1. *QOS-AWARE SERVICE SELECTION is solvable in linear time for instances with only one criterion.*

Theorem 2. *An instance of QOS-AWARE SERVICE SELECTION on p criteria, $p \geq 2$, with one of them of cat 4 or cat 5, can be reduced in polynomial time to several instances on $p - 1$ criteria.*

Corollary 1. QOS-AWARE SERVICE SELECTION is polynomial time solvable for instances with two criteria where one of them is of cat 4 or cat 5.

Theorem 3. For a fixed number of criteria p , QOS-AWARE SERVICE SELECTION is solvable in pseudo-polynomial time.

4 Complexity Lower Bounds

In this section, we prove that the positive results of the previous section are tight, i.e. QOS-AWARE SERVICE SELECTION is weakly NP-hard. We will show that it is the case even for very restricted instances.

For an instance of QOS-AWARE SERVICE SELECTION with two criteria c_1 and c_2 we denote by MAX/MAX (or MIN/MIN) when c_1 and c_2 are to be maximized (or minimized) and by MIN/MAX (resp. MAX/MIN) when c_1 is to be minimized (resp. maximized) and c_2 to be maximized (resp. minimized).

Theorem 4. When $p = 2$ and the criteria c_1 and c_2 are both of cat 1 or both of cat 3, then QOS-AWARE SERVICE SELECTION is NP-hard even when

1. one AND pattern, 2 services per task, and MAX/MAX (or MIN/MIN, or MIN/MAX).
2. one sequence pattern, 2 services per task, and MAX/MAX (or MIN/MIN, or MIN/MAX).
3. a sequence of XOR patterns, 1 service per task, and MAX/MAX (or MIN/MIN, or MIN/MAX).
4. one OR pattern, 1 service per task, and MIN/MAX.

5 Conclusion

In this work, we established NP-hardness proofs for QOS-AWARE SERVICE SELECTION, which hold even in very restrictive cases. However, we show that this hardness result is counter-balanced by the existence of a pseudo-polynomial time algorithm, which is able to solve optimally the problem in polynomial time for small values of QoS services, which is the common case in real instances.

For future work, we suggest investigating the complexity of the problem in case of two criteria of cat 1 and cat 3. However, one would easily notice that if there are three criteria (at least two of cat 1, or at least two of cat 3), then the problem is clearly NP-hard due to Theorem 4. It would also be interesting to investigate the potential relation between the shape of a workflow and the complexity of the problem. Of particular interest are cases where the problem becomes strongly NP-hard. Another interesting direction would be to establish approximation algorithms that return solutions with a priori guarantee of quality in polynomial time even for large values of QoS services.

Acknowledgements. We are grateful for the support by the bilateral research cooperation CEDRE between France and Lebanon (grant number 30885TM).

References

1. Alrifai, M., Risse, T., Nejdl, W.: A hybrid approach for efficient web service composition with end-to-end QoS constraints. *ACM Trans. Web* **6**(2), 7:1–7:31 (2012)
2. Ardagna, D., Pernici, B.: Global and local QoS guarantee in web service selection. In: Bussler, C.J., Haller, A. (eds.) *BPM 2005*. LNCS, vol. 3812, pp. 32–46. Springer, Heidelberg (2006)
3. Bonatti, P.A., Festa, P.: On optimal service selection. In: *Proceedings of the 14th International Conference on World Wide Web*, pp. 530–538 (2005)
4. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for QoS-aware service composition based on genetic algorithms. In: *Proceedings of the 7th GECCO*, pp. 1069–1075. ACM (2005)
5. El Haddad, J.: Optimization techniques for QoS-aware workflow realization in web services context. In: Lacroix, Z., Vidal, M.E. (eds.) *RED 2010*. LNCS, vol. 6799, pp. 134–149. Springer, Heidelberg (2012)
6. Gabrel, V., Manouvrier, M., Murat, C.: Web services composition: complexity and models. *Discrete Appl. Math.* (2014). <http://dx.doi.org/10.1016/j.dam.2014.10.020>
7. Kiepuszewski, B., ter Hofstede, A.H.M., Bussler, C.J.: On structured workflow modelling. In: Wangler, B., Bergman, L.D. (eds.) *CAiSE 2000*. LNCS, vol. 1789, pp. 431–445. Springer, Heidelberg (2000)
8. Moghaddam, M., Davis, J.G.: Service selection in web service composition: a comparative review of existing approaches. In: Bouguettaya, A., Sheng, Q.Z., Daniel, F. (eds.) *Web Services Foundations*, pp. 321–346. Springer, New York (2014)
9. Schuller, D., Miede, A., Eckert, J., Lampe, U., Papageorgiou, A., Steinmetz, R.: QoS-based optimization of service compositions for complex workflows. In: Fantinato, M., Yang, J., Weske, M., Maglio, P.P. (eds.) *ICSOC 2010*. LNCS, vol. 6470, pp. 641–648. Springer, Heidelberg (2010)
10. Schuller, D., Polyvyanyy, A., García-Bañuelos, L., Schulte, S.: Optimization of complex QoS-aware service compositions. In: Kappel, G., Motahari-Nezhad, H.R., Maamar, Z. (eds.) *Service Oriented Computing*. LNCS, vol. 7084, pp. 452–466. Springer, Heidelberg (2011)
11. Strunk, A.: QoS-aware service composition: a survey. In: *2010 IEEE 8th European Conference on Web Services (ECOWS)*, pp. 67–74. IEEE (2010)
12. Trummer, I., Faltings, B., Binder, W.: Multi-objective quality-driven service selection - a fully polynomial time approximation scheme. *IEEE Trans. Softw. Eng.* **40**(2), 167–191 (2014)
13. van Der Aalst, W.M., Ter Hofstede, A.H., Kiepuszewski, B., Barros, A.P.: Workflow patterns. *Distrib. Parallel Databases* **14**(1), 5–51 (2003)
14. Yu, T., Lin, K.-J.: Service selection algorithms for web services with end-to-end QoS constraints. *Inf. Syst. E-Business. Manage* **3**(2), 103–126 (2005)
15. Zeng, L., Benatallah, B., Ngu, A.H.H.: QoS-aware middleware for web services composition. *IEEE Trans. Softw. Eng.* **30**, 311–327 (2004)
16. Zhang, W., Chang, C.K., Feng, T., Jiang, H.: QoS-based dynamic web service composition with ant colony optimization. In: *34th Annual Computer Software and Applications Conference*, pp. 493–502. IEEE (2010)

TRACE: A Dynamic Model of Trust for People-Driven Service Engagements

Combining Trust with Risk, Commitments, and Emotions

Anup K. Kalia^(✉), Pradeep K. Murukannaiah,
and Munindar P. Singh

North Carolina State University, Raleigh, NC 27695-8206, USA
{akkalia, pmuruka, singh}@ncsu.edu

Abstract. Trust is an important element of achieving secure collaboration that deals with human judgment and decision making. We consider trust as it arises in and influences people-driven service engagements. Existing approaches for estimating trust between people suffer from two important limitations. One, they consider only *commitment* as the primary means of estimating trust and omit additional significant factors, especially *risk* and *emotions*. Two, they typically estimate trust based either on fixed parameter models that require manual setting of parameters or based on Hidden Markov Models (HMM), which assume conditional independence and are thus ill-suited to capturing complex relationships between trust, risk, commitments, and emotions.

We propose TRACE, a model based on Conditional Random Fields (CRF) that predicts trust from risk, commitments, and emotions. TRACE does not require manual parameter tuning and relaxes conditional independence assumptions among input variables. We evaluate TRACE on a dataset collected by the Intelligence Advanced Research Projects Activity (IARPA) in a human-subject study. We find that TRACE outperforms existing trust-estimation approaches and that incorporating risk, commitments, and emotions yields lower trust prediction error than incorporating commitments alone.

1 Introduction

People-driven service engagements involve how people interact to carry out collaborative business processes [6, 7]. Such business processes involve human judgment and decision-making [13]. Trust is a crucial element of achieving secure collaboration where people interact since it enhances the quality of a collaboration. We consider direct interaction between people, which can be used to inform the design of user agents to facilitate collaboration among people. As people interact, they estimate and continually revise trust in each other based on their mutual interactions. Trust is established as a crucial element of service selection, e.g., [11]. However, as service settings become more complex and intertwined with social interactions, we need to expand our understanding of trust in services to promote the human element.

Existing approaches to trust estimation consider commitments alone. Gambetta [4] interprets trust as a truster’s assessment of a trustee for performing a specific task. Mayer et al. [12] define trust as the willingness of a truster to be vulnerable to a trustee for the completion of a task. Teacy et al. [18] consider trust as the truster’s estimation of probability that a trustee will fulfill its obligation toward the truster. Wang et al. [20] represent trust as the belief of a truster that the trustee will cooperate, and estimate trust by aggregating positive and negative experiences. Singh [16] provides a formal semantics for trust that supports various postulates on trust, including how trust relates to commitments. Kalia et al. [8] consider commitments to predict trust.

Considering trust as a dynamic variable, two major classes of trust models arise in the literature. First, fixed-parameter trust models, where the parameters of the model are manually fixed, typically, based on heuristics [18,20]. Second, machine-learned trust models, typically Hidden Markov Models (HMM) [10,19,21], assume that input variables are conditionally independent of each other given the output variable.

Research Question. Our overarching question is: How can we improve trust prediction by incorporating (in addition to commitments) two attributes (1) *risk* taken by a truster toward a trustee, and (2) *emotions* displayed by a truster toward a trustee without presuming conditional independence? We consider risk because it depends on a truster’s belief about the likelihood of gains or losses it might incur from its relationship with a trustee [12]. For example, a manager may trust a subordinate who performs a high-risk task more than another who performs a low-risk task, even if both subordinates succeed at the task. Conversely, the manager may assign high-risk tasks to a subordinate whom he or she trusts more than the other. We consider emotions because studies in psychology suggest that positive emotions (e.g., happiness, gratitude) increase trust, whereas negative emotions (e.g., anger) decrease trust [2]. Conditional independence may not hold in our setting. For example, consider the relationships between trust (output variable) and risk and commitments (two input variables). An HMM model would assume that risk and commitments are independent given the level of trust. However, the likelihood of gaining from risk is higher if commitments are satisfied.

Contributions. We propose TRACE, a model of trust based on Conditional Random Fields (CRF) [9]. TRACE avoids manual fixing of parameters and relaxes the conditional independence assumption. To create TRACE, first, we propose relationships between trust, risk, commitments, and emotions. Then, we train TRACE using the past observations between people. Once TRACE is trained, we use it to infer trust given new observations. Our claims are two fold: (1) by capturing complex relationships among output and input variables, TRACE estimates trust between people better than fixed-parameter and HMM-based trust models, and (2) by capturing risk, commitments, and emotions, TRACE performs better than models that capture only commitments. We evaluate our claims via data collected from a human-subject study conducted by the Intelligence Advanced Research Projects Activity (IARPA).

2 A Conceptual Model of Trust

TRACE enhances Mayer et al.'s [12] trust antecedent framework (TAF) as shown in Fig. 1. The trust model contains four variables: trust (T), risk (R), commitments (C), and emotions (E). We describe each variable $V = \langle T, R, C, E \rangle$ using Singh's [15, 16] formal notation $V \langle \text{debtor}, \text{creditor}, \text{antecedent}, \text{consequent} \rangle$. In the notation, the debtor and the creditor are the roles enacted by individuals. The antecedent represents *conditions* and the consequent represents *tasks*.

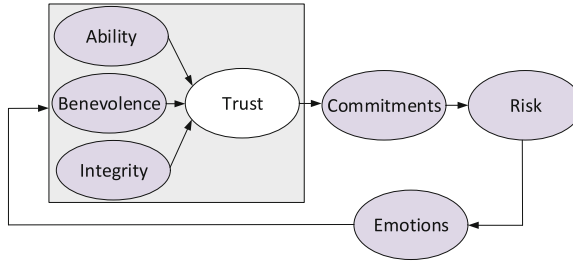


Fig. 1. The TRACE model enhances the trust antecedent framework [12] with emotion.

Commitments. We represent a commitment as $C \langle \text{trustee}, \text{truster}, \text{antecedent}, \text{consequent} \rangle$. In a commitment, the trustee commits to the truster to perform the consequent. If the trustee performs the consequent, the commitment is *satisfied*. If the antecedent is true but the trustee does not perform the consequent, the commitment is *violated*.

Risk. We represent risk as $R \langle \text{truster}, \text{trustee}, \text{antecedent}, \text{consequent} \rangle$, denoting that the truster takes a risk by accepting the trustee's offer to perform the consequent. If the trustee performs the consequent, the truster *gains*; else, the truster suffers a *loss*.

Trust. We represent trust as $T \langle \text{truster}, \text{trustee}, \text{antecedent}, \text{consequent} \rangle$, denoting that the truster *believes* the trustee if the trustee performs the consequent. If the trustee does not perform the consequent, the truster begins to *doubt* the trustee. Based on TAF, trust has three dimensions: (1) *ability*, the trustee's competency to perform the consequent, (2) *benevolence*, the trustee's willingness to perform the consequent, and (3) *integrity*, the trustee's ethics and morality in performing the consequent.

Emotions. An emotion is a psychological response to an external or internal event [3, 17]. We introduce emotions as a response to commitment outcomes (satisfaction or violation) in the TAF (Fig. 1). Similar to trust and risk, we denote emotions as $E \langle \text{truster}, \text{trustee}, \text{antecedent}, \text{consequent} \rangle$. The truster displays a *positive emotion* if the trustee performs the consequent, else a *negative emotion*.

Postulates. Next, we propose postulates that capture relationships between the variables above. In these postulates, V_t represents the state of the variable V at time t .

- P₁**: $T_t \rightarrow T_{t+1}$. The trust T_{t+1} is influenced by the past trust T_t . This postulate is consistent with the HMM trust models [10,19,21] since they assume that a truster computes its current trust T_{t+1} for a trustee based on its past trust T_t with the trustee.
- P₂**: $C_t \rightarrow T_t$. The current commitment outcome C_t influences the current trust T_t . We consider this postulate since Kalia et al. [8] suggest that a truster trusts a trustee if the trustee satisfies the trustee’s commitments toward the truster.
- P₃**: $R_t \rightarrow C_t$. The risk taken influences the commitment outcome C_t or the gain or loss realized in the risk R_t . The postulate is supported by TAF [12].
- P₄**: $R_t \rightarrow T_t$. The current risk taken R_t influences the current trust T_t . The postulate is supported by TAF [12].
- P₅**: $C_t \rightarrow E_t$. The commitment outcome C_t influences the current emotion E_t . Smith and Ellsworth [17] suggest that a truster’s emotions depend on the truster’s appraisal of a trustee’s commitments toward the truster.
- P₆**: $R_t \rightarrow E_t$. The risk taken R_t influences the truster’s emotion E_t . We consider this postulate to capture the indirect effect that the risk taken influences the commitment outcomes (P₃), which influence emotions (P₅).
- P₇**: $E_t \rightarrow T_t$. The current emotion E_t influences the current trust T_t . Psychological studies suggest that a truster makes trust-based judgments toward a trustee based on his or her emotional relationships with the trustee [2].

3 The TRACE Model

To compute trust, we propose the TRACE model using dynamic Bayesian models. In these models, we consider T, R, C, and E as random variables. Using the variables and the relationships proposed above, we construct two dynamic Bayesian models as show in Figs. 2a and b, respectively. Figure 2a represents the HMM model (the state-of-the-art-model) whereas Fig. 2b represents the TRACE model.

HMM-Based Solutions and their Limitations. Dynamic Bayesian models such as HMMs can be adapted to compute trust as shown in Fig. 2a. Here, input variables are considered as a sequence of observations $\mathbf{x}=\{C, R, E\}_{t=1}^T$, and output variables are considered as a sequence of states $\mathbf{y}=\{T\}_{t=1}^T$ where T is the length of a specific sequence. Then, a HMM represents the joint distribution $p(\mathbf{y}, \mathbf{x})$, making two independence assumptions: (1) the current state y_t is independent of y_1, y_2, \dots, y_{t-2} , given y_{t-1} ; (2) observations x_t are independent of each other, given y_t . Given these independence assumptions, the joint distribution can be computed as $p(\mathbf{y}, \mathbf{x}) = \prod_{t=1}^T p(y_t|y_{t-1}) \times p(x_t|y_t)$. However, a downside of making these assumptions is that the corresponding models ignore some of the trust dependencies postulated in Sect. 2. For example, the HMM shown in Fig. 2a assumes C^t to be independent of E^t given trust T^t , which may not be true according postulate P₅.

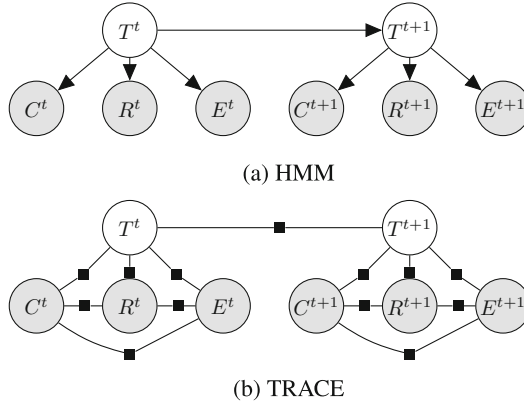


Fig. 2. Graphical representation of HMM and TRACE trust models (two time slices).

TRACE. TRACE employs CRFs to overcome the limitations of HMM-based trust models. As shown in Fig. 2b, our CRF-based model considers all the dependencies postulated in Sect. 2. As Lafferty et al. [9] describe, unlike HMMs, CRFs are agnostic to dependencies between the observations. Further, the conditional probability of the label sequence can depend on arbitrary, nonindependent features of the observation sequence without forcing the model to account for the distribution of those dependencies. CRFs capture relationships between input and output variables (\mathbf{x}, \mathbf{y}) as *feature functions* (undirected edges in the graphical model shown in Fig. 2b). A feature function can be computed by considering the entire input sequence.

An HMM model simplifies the computation of the joint probability by assuming conditional independence. In contrast, a CRF model employs discriminative modeling, where the distribution $p(\mathbf{y}|\mathbf{x})$ is learned directly from the data (not requiring to learn the parameters of the entire joint distribution). The most important aspect of CRFs is to relate $p(\mathbf{y}|\mathbf{x})$ and feature functions $f_k(y_t, y_{t-1}, x_t)$. Each feature function covers either a state-state pair (y_t, y_{t-1}) , e.g., (T_{t+1}, T_t) or a state-observation pair (x_t, y_t) , e.g., (C_t, T_t) , (E_t, T_t) , and (R_t, T_t) . Suppose we have K feature functions that represent state-state and state-observation pairs from \mathbf{x} and \mathbf{y} . Then, $p(\mathbf{y}|\mathbf{x})$ can be computed starting from the joint distribution $p(\mathbf{y}, \mathbf{x})$ as follows.

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{y}, \mathbf{x})}{\sum_{\mathbf{y}} p(\mathbf{y}, \mathbf{x})} = \frac{\exp \left\{ \sum_{k=1}^K \lambda_k f_k(y_t, y_{t-1}, x_t) \right\}}{\sum_{\mathbf{y}} \exp \left\{ \sum_{k=1}^K \lambda_k f_k(y_t, y_{t-1}, x_t) \right\}}. \quad (1)$$

Training. To estimate the parameters λ_k in Eq. 1, we consider the training data $\mathcal{D} = \{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^N$. The parameters can be estimated by maximizing the log-likelihood \mathcal{L} on the training data \mathcal{D} , i.e., $\mathcal{L}(\mathcal{D}) = \sum_{i=1}^N \log p(\mathbf{y}|\mathbf{x})$.

Inference. To find the best possible state sequence \mathbf{y} for observations \mathbf{x} , we use the Viterbi algorithm [14]. According to the algorithm, we define a quantity $\delta_t(i)$ that indicates the highest score (highest probability) of a path at time t as $\delta_t(i) = \max_{y_1, y_2, \dots, y_{t-1}} p(y_1, y_2, \dots, y_t = i, x_1, x_2, \dots, x_t | \lambda)$ where i represents the state at time t .

4 Evaluation

We evaluate TRACE on data collected from subjects executing the Checkmate protocol [5] adapted from the iterated investment or dictator economic decision-making game [1]. The subjects assessed each other's trustworthiness as they played the game.

The Checkmate Protocol. The protocol involves two roles: *banker* and *game player*. The banker's task is to loan money to a game player from an initial endowment of 50 USD. The game player's task, in a single round of the protocol, is to complete a virtual maze of desired difficulty and collect as many cash boxes hidden in the maze as possible within the allotted time. The game player requests a loan from the banker to play a maze, promising to play a maze of certain difficulty and return (1) the loan with all gains, (2) the loan with 50% of all gains, (3) 50% of the available money, or (4) a fixed amount. After the game player's request, the banker chooses a loan category: small (1–7 USD), medium (4–10 USD), or big (7–13 USD). Then, a dollar amount, randomly generated within the banker's chosen category, is loaned to the game player. The game player does not know the category chosen by the banker. Next, the game player plays a maze of a certain difficulty (not necessarily what he or she had promised). The banker will not know the actual maze played. The difficulty of the maze determines the risk involved: low risk (75–150%; i.e., the player could lose up to 25% or gain up to 150% of the loan amount in this maze), moderate risk (50–200%), or high risk (0–300%). Finally, the game player returns some money to the banker (not necessarily what he or she had promised).

A pair of subjects (one banker; one player) executed the protocol for up to five rounds. After each round, the subjects answered questions about their (individual) emotions and perceptions of the opponent's trustworthiness. All the money involved was real—that is, subjects kept the money they were left with at the end of all rounds.

Data. The data consists of 431 rows collected from 63 subjects, where each row corresponds to the sequence of rounds played between two subjects. The data we obtained reflects only the banker's perspective. Thus, our observations and predictions are from the banker's perspective. We compute the variables of our interest for each round in a sequence as follows. (1) We treat the commitment from the player to the banker, $C\langle \text{player}, \text{banker}, \text{loan}, \text{return} \rangle$, as satisfied if the player returned at least the amount he or she had loaned, and as violated, otherwise. (2) We compute the gain or loss in the risk, $R\langle \text{banker}, \text{player}, \text{loan}, \text{return} \rangle$, based on the difference between the loaned and returned amounts. (3) The dataset represents the banker's trust for the player after the round, $T\langle \text{banker}, \text{player}, \text{loan}$,

return), as a three-tuple $\langle A, B, I \rangle$, indicating the banker’s perception of player’s ability, benevolence, and integrity, respectively, each a real value (0–1) derived from the post-round questionnaire. (4) The dataset represents the banker’s emotion after he or she receives a return from the player, $E\langle \textit{banker}, \textit{player}, \textit{loan}, \textit{return} \rangle$, as real-valued (1–10) state anxiety scores derived from the post-round questionnaire.

Mean Absolute Error (MAE). We treat trust estimation as a classification problem. Thus, we discretized each trust dimension (A, B, and I) into three categories (low, medium, and high) of almost equal frequency, making sure that no trust value is repeated across categories. The sizes of the resulting categories were A: [114, 155, 162], B: [109, 163, 159], and I: [118, 136, 177]. We measure the performance of a trust model via $MAE = \frac{\sum_i^N |actual^i - predicted^i|}{N}$.

Comparison. For comparing HMM and TRACE, we perform a three-fold cross validation and compare the average MAE of the three folds. For each model, we considered the following feature combinations to predict trust: (1) C: only commitments, (2) C+R: commitments and risk, (3) C+E: commitments and emotions, (4) R+E: risk and emotions, and (5) C+R+E: commitments, risk and emotions. For each of these settings, we hypothesize that TRACE yields a lower MAE than HMM.

5 Results and Discussion

Table 1 compares HMM and TRACE, considering different feature combinations for predicting trust. When we consider only C, TRACE yields lower MAEs than HMM for each trust attribute. The primary reason for the result might be that CRF employs discriminative modeling whereas HMM employs generative modeling. Considering all features (C + R + E), TRACE again yields lower MAEs than HMM for each trust attribute (A, B, and I). We attribute this result to dependencies between C, R, and E, given T, which HMM ignores but TRACE incorporates.

Next, for C + R, TRACE performs better than HMM in predicting A and I (MAEs for B are quite similar). Thus, not assuming C and R as conditionally independent given T (as TRACE does) is beneficial to trust prediction than assuming so (as HMM does). However, for C + E, HMM performs better than TRACE for B and I (MAEs for A are quite similar). Thus, treating C and E as conditionally independent is beneficial to trust prediction than not treating so. This result suggests that changes in a truster’s emotions are not limited to the appraisal of commitments, but can depend on other factors present in the truster’s environment [17]. The result for R + E is mixed: TRACE performs better than HMM for B and I, whereas HMM performs better than TRACE for A.

In summary, these observations suggest that dependency relationships between commitments, risk, and emotions vary depending on whether the observed trust

Table 1. MAEs of HMM and TRACE considering different feature combinations.

Input variables	HMM			TRACE		
	A	B	I	A	B	I
C	1.1220	0.8564	1.0917	0.8744	0.7576	0.7988
C + R	0.8974	0.7655	0.8484	0.7463	0.7685	0.7876
C + E	0.8619	0.7433	0.7184	0.8617	0.7656	0.7580
R + E	0.8468	0.8376	0.7992	0.8949	0.6815	0.6568
C + R + E	0.8870	0.7977	0.7714	0.7878	0.7427	0.7141

attribute is ability, benevolence, or integrity. Thus, our finding can be valuable in choosing the right set of dependencies given input and output variables of interest.

Threats to Validity. We identify three caveats about our evaluation. First, our dataset, although real, consists of short sequences. We expect both HMM and TRACE to perform better given longer sequences. Second, the dataset is skewed toward positive trust values and our conclusions may not hold since the trust values have a different distribution. Third, the dataset represents emotions using anxiety scores only, thereby lacking realistic emotion responses along multiple dimensions such as anger and joy.

Discussion. Despite these limitations, TRACE illustrates that a probabilistic model of trust that incorporates commitments, risk, and emotions can produce trust estimates with fairly good accuracy. Collaboration inevitably involves one party making itself vulnerable to another and inherently involves negotiation. The negotiation may be explicit, as in the dataset we studied, or implicit, such as when one party decides whether to take up any offer from another, including commonplace situations such as accessing a weblink or an email attachment. Our findings therefore open up the possibility of developing user agents that promote secure collaboration by helping a user calibrate the perceived trust with the risk undertaken in light of available measures of risk and gain from commitments. We defer investigating such agents to future research.

Acknowledgments. Thanks to the NCSU Laboratory of Analytic Sciences and to the US Department of Defense for support through the Science of Security Lablet. Thanks to Zhe Zhang, Chung-Wei Hang, and the anonymous reviewers for useful comments.

References

1. Berg, J., Dickhaut, J., McCabe, K.: Trust, reciprocity, and social history. *Games Econ. Behav.* **10**(1), 122–142 (1995)
2. Dunn, J.R., Schweitzer, M.E.: Feeling and believing: the influence of emotion on trust. *J. Pers. Soc. Psychol.* **88**(5), 736–748 (2005)
3. Friedenberg, J., Silverman, G.: *Cognitive Science*, 2nd edn. SAGE Publications, Thousand Oaks (2012)

4. Gambetta, D.: Can we trust trust? In: *Trust: Making and Breaking Cooperative Relations*. Blackwell, New York (1988)
5. IARPA: The checkmate protocol (2014). <https://www.innocentive.com/ar/challenge/9933465>
6. Kalia, A.K., Motahari-Nezhad, H.R., Bartolini, C., Singh, M.P.: Monitoring commitments in people-driven service engagements. In: *Proceedings of the 10th IEEE International Conference Service Computing*, pp. 160–167 (2013)
7. Kalia, A.K., Singh, M.P.: Muon: designing multiagent communication protocols from interaction scenarios. *J. Auton. Agents Multi-Agent Syst.* **29**(4), 621–657 (2015)
8. Kalia, A.K., Zhang, Z., Singh, M.P.: Estimating trust from agents' interactions via commitments. In: *Proceedings of the 21st European Conference Artificial Intelligence*, pp. 1043–1044 (2014)
9. Lafferty, J.D., McCallum, A.K., Pereira, F.C.N.: Conditional random fields. In: *Proceedings of the 18th International Conference on Machine Learning*, pp. 282–289 (2001)
10. Liu, X., Datta, A.: Modeling context aware dynamic trust using hidden markov model. In: *Proceedings of the 26th National Conference on Artificial Intelligence*, pp. 1938–1944 (2012)
11. Maximilien, E.M., Singh, M.P.: Toward autonomic web services trust and selection. In: *Proceedings of the 2nd International Conference on Service-Oriented Computing*, pp. 212–221 (2004)
12. Mayer, R.C., Davis, J.H., Schoorman, F.D.: An integrative model of organizational trust. *Acad. Manag. Rev.* **20**(3), 709–734 (1995)
13. Nezhad Motahari, H.R., Spence, S., Bartolini, C., Graupner, S., Bess, C., Hickey, M., Joshi, P., Mirizzi, R., Ozonat, K., Rahmouni, M.: Casebook: a cloud-based system of engagement for case management. *IEEE Internet Comput.* **17**(5), 30–38 (2013)
14. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. In: *Readings in Speech Recognition*, pp. 267–296. Morgan Kaufmann, San Mateo (1990)
15. Singh, M.P.: An ontology for commitments in multiagent systems: toward a unification of normative concepts. *Artif. Intell. Law* **7**(1), 97–113 (1999)
16. Singh, M.P.: Trust as dependence: a logical approach. In: *Proceedings of the 10th International Conference on Autonomous Agents and MultiAgent System*, pp. 863–870, Taipei (2011)
17. Smith, C.A., Ellsworth, P.C.: Patterns of cognitive appraisal in emotion. *J. Pers. Soc. Psychol.* **48**(4), 813–838 (1985)
18. Teacy, W.L., Patel, J., Jennings, N.R., Luck, M.: Travos: trust and reputation in the context of inaccurate information sources. *J. Auton. Agents Multi-Agent Syst.* **12**(2), 183–198 (2006)
19. Vogiatzis, G., MacGillivray, I., Chli, M.: A probabilistic model for trust and reputation. In: *Proceedings of the 9th International Conference Autonomous Agents and Multiagent System*, pp. 225–232 (2010)
20. Wang, Y., Hang, C.W., Singh, M.P.: A probabilistic approach for maintaining trust based on evidence. *J. Artif. Intell. Res.* **40**, 221–267 (2011)
21. Zheng, X., Wang, Y., Orgun, M.A.: Modeling the dynamic trust of online service providers using HMM. In: *Proceedings of the 20th IEEE International Conference on Web Services*, pp. 459–466 (2013)

A Context-Aware Approach for Personalised and Adaptive QoS Assessments

Lina Barakat^(✉), Adel Taweel, Michael Luck, and Simon Miles

Department of Informatics, King's College London, London, UK
{lina.barakat,adel.taweel,michael.luck,simon.miles}@kcl.ac.uk

Abstract. Given the importance of QoS (quality of service) properties for distinguishing between functionally-equivalent services and accommodating different user expectations, a number of QoS estimation approaches have been proposed, utilising the observation history available on a service. Although the context underlying such previous observations (and corresponding to both user and service related factors) could provide an important source of information for the QoS estimation process, it has only been utilised to a limited extent by existing approaches. In response, we propose a context-aware quality learning model, realised via a learning-enabled service agent, exploiting the contextual characteristics of the domain in order to provide more personalised, accurate and relevant quality estimations for the situation at hand. The experiments conducted demonstrate the effectiveness of the proposed approach.

Keywords: Context awareness · Change detection · Personalisation · Quality value learning

1 Introduction

Services advertised by different providers can overlap in their functional capabilities, but offer varying quality of service (QoS) levels. Such QoS properties, thus, play an essential role in differentiating between functionally equivalent services and accommodating different user needs. However, the subjective nature of some properties and the dynamic and unreliable nature of service environments may result in cases where the quality values available from the service provider are either uninstantiated or untrustworthy. Consequently, a number of efforts focus on learning more accurate estimation of service quality values, based on the data available regarding the service's past performance (e.g. [5, 7, 8]). Most such learning approaches, however, rely on data recency to account for potential changes in the service's behaviour. That is, newer service observations are favoured, while older ones are eventually forgotten, without consideration of the service's circumstances, thus neglecting important evidence for detecting a change occurrence. Moreover, the observations are usually assumed to be objective, and thus the predictions produced do not account for a user's particular situation. We argue

that accounting for the circumstances under which the observations were collected (in relation to both the user and the service) is essential to ensure that only relevant data is captured in the learning process, as illustrated below.

User Circumstances. Consider a scenario where a user wants to order a meal for dinner, and therefore contacts a food-specialised broker. The broker has access to information about a pool of meal delivery services that are offered by various food providers. Let's assume that the user suffers from chewing and swallowing problems, and therefore requires the meal to be of tender texture. Given a candidate meal option, the broker thus needs to assess its corresponding texture from past available ratings to determine its suitability for the user. Since the perception of food texture could be affected by the presence of chewing and swallowing difficulties, the ratings of users sharing similar dysphagia conditions with the current user should have the highest impact on texture assessment at hand, while the contribution of those with no difficulties should be minimal.

Service Circumstances. Consider a similar food ordering scenario where a user is interested in a meal that is highly rated in terms of taste. Again, the broker here needs to assess the taste property of each candidate meal option. Assume one such option is service s , with a good rating history up to time step t_k , after which the service exhibits a change in recipe, occurring, for instance, due to a change in the head chef, or the temporary unavailability of some ingredients (e.g. some ingredients might not be available at winter time). Such a change could affect many aspects of the meal, including taste, making such previous user observations under the old recipe less (or no longer) relevant under the new one. Now, if the service switches again to the old recipe, window $[t_1, t_k]$ of the historical observations on taste available for service s becomes relevant again, and is a useful source of information for assessment of taste for this service.

Given this, we propose enriching service observations with contextual information, and exploiting such information during QoS learning to capture the most relevant data for the situation at hand, thus achieving more personalised and adaptive quality predictions. By context, we refer to any conditions and circumstances that may affect the perception of a quality value by a service user, either related to the user itself (user context) or related to the service (service context). The context model is presented in Sect. 2. Sections 3 and 4 present our context-aware QoS learning model and the experimental results, respectively. Related work and conclusion are discussed in Sects. 5 and 6.

2 Context Model

The quality characteristics of a service may be dependent on the user situation (user context) under which the service provision happens (e.g. user's location), as well as on service-related circumstances (service context), which could change over time either periodically (e.g. a change in a food service's recipe with season) or non-periodically (e.g. a rare event such as a sudden server crash).

Formally, knowledge of context information relevant for a service provision is a tuple $(Q, C^u, C^s, ctx_u, ctx_s, dom)$, as follows. Q is the set of quality of service attributes of the service, either generic such as price and response time, or domain-dependent such as taste of a food service. C^u and C^s are the sets of attributes characterising a user's context and the service's context, respectively, expected to affect the quality values delivered by the service (and are shared among similar service types). ctx_u is a quality attribute's user context function, mapping quality attribute $q \in Q$ to the user-related context attributes $ctx_u(q) \subset C^u$ that may have an impact on the perception of q by the user, e.g., $ctx_u(\text{texture}) = \{\text{chewing and swallowing condition}\}$, and $ctx_u(\text{price}) = \emptyset$. ctx_s is a quality attribute's service context function, mapping quality attribute $q \in Q$ to the service-related context attributes $ctx_s(q) \subset C^s$ that may have an impact on the behaviour of the service so that the same user may observe different values of q under different values of these attributes, e.g., $ctx_s(\text{taste}) = \{\text{food recipe}\}$. Finally, dom is an attribute domain function, mapping an attribute a (quality or contextual) to its corresponding set of possible values. In this paper, we assume that $dom(a)$ corresponds to a discrete domain (for continuous attributes, this is obtained via applying an appropriate discretisation algorithm).

3 Context-Aware QoS Learning

In our approach, a service's QoS characteristics are assessed via a learning-enabled agent associated with the service. This agent (which, for example, could be acting on behalf of the service provider or the broker with which the service is registered) exploits a contextually-enriched history of past interactions with the service in order to expose personalised and dynamism-aware QoS information to clients. The modelling and learning details of such an agent are presented next.

3.1 Service Observation

For each quality attribute $q \in Q$, the agent receives a stream of service observations, each reporting the outcome encountered for q in a previous interaction with the service, along with the contextual circumstances surrounding this interaction. Formally, a service observation is denoted as $(v_q, \mathbf{v}_u, \mathbf{v}_s)$, where: $v_q \in dom(q)$ is the value observed for q ; $\mathbf{v}_u = (v_u^1, \dots, v_u^m)$ are the respective values of user-side contextual attributes $(c_u^1, \dots, c_u^m) \in ctx_u(q)^m$; and $\mathbf{v}_s = (v_s^1, \dots, v_s^k)$ are the respective values of service-side contextual attributes $(c_s^1, \dots, c_s^k) \in ctx_s(q)^k$.

3.2 Agent Configuration and Learning Model

The main idea behind our approach is that, for a particular quality attribute, the agent maintains a set of learned *value models*, each corresponding to a different behaviour of the service (as a result of changes in service-side circumstances). When previously-encountered service circumstances reoccur, older observations of the service collected under such circumstances become relevant again, and

the agent can reuse the respective historical value model (learned from these observations) to make future quality value predictions. Such reuse of a previously learned value model facilitates a faster adaptation to a behavioural change of the service (as opposed to re-learning the behaviour from new interactions), and consequently improves the accuracy of quality predictions.

Formally, the configuration of a service agent at a particular time step is a tuple $(\Omega, active)$, where: Ω is the model library of the agent, containing the set of learned value models for quality attributes; and function *active* maps each quality attribute $q \in Q$ to its currently active value model $active(q) \in \Omega$ (i.e. the model utilised to predict the attribute's value for the next discovery attempt by a consumer). Each model $\omega \in \Omega$ is of the form $q : \psi \leftarrow M$. Here, $q \in Q$ is the quality attribute the value of which the model is trying to predict. Precondition ψ identifies the service-side contextual circumstances under which the model is valid (it is a logical formula in disjunctive normal form (DNF) restricting the values of contextual attributes $c_s \in ctx_s(q)$). Finally, body M is the actual prediction model for quality attribute q under condition ψ . It corresponds to the underlying function *qual* between the values of user-side contextual attributes affecting q and the corresponding value of q , i.e. $qual(q, \mathbf{v}_u) \in dom(q)$ is the value predicted for quality attribute q given user's contextual values \mathbf{v}_u .

The configuration of the agent evolves over time as the agent's learning progresses. In particular, given the current configuration $(\Omega, active)$, and a new service observation $(v_q, \mathbf{v}_u, \mathbf{v}_s)$, value vector \mathbf{v}_s is compared against the contextual precondition ψ of the currently active model $active(q)$, and two cases are distinguished. Case 1. \mathbf{v}_s is subsumed by ψ , in which case no behavioural drift is assumed, and observation (v_q, \mathbf{v}_u) is simply used to update body M of the currently active model $active(q)$ to increase its accuracy with more incoming data. Case 2. \mathbf{v}_s is not subsumed by ψ , in which case a behavioural drift is suspected and further two sub-cases are distinguished. Case 2.1. \mathbf{v}_s is subsumed by the contextual precondition ψ' of another existing model $\omega' \in \Omega$ of attribute q ($\omega' \neq active(q)$). Here, a recurring behaviour (i.e. a behaviour learned previously) is assumed, and the respective model ω' becomes the current active model for attribute q , with its body M' being updated with observation (v_q, \mathbf{v}_u) . Case 2.2. \mathbf{v}_s is not subsumed by any contextual precondition of any previously learned model for quality attribute q . In this case, the agent suspects a new service behaviour, and therefore set up a new model ω^n for attribute q , which is added to the model library. The contextual precondition ψ^n of this model is the conjunction of values \mathbf{v}_s , while its body M^n is built incrementally from the new incoming observations starting from the current observation (v_q, \mathbf{v}_u) .

After the new model ω^n is stabilised (i.e. after *stability* incoming observations under condition ψ^n), it is compared against the other existing models in the agent's library to verify whether it is actually reflecting a new service behaviour for attribute q (we utilise the *conceptual equivalence measure* proposed by Yang et al. [1] for such comparison). If no similar model is found, the new behaviour is confirmed and model ω^n becomes the currently active model for attribute q . Otherwise (i.e. a similar model ω^{sim} exists in the library), model ω^n is discarded

(i.e. removed from the library), while model ω^{sim} is regarded as the currently active model for q , with its contextual precondition ψ^{sim} being generalised to subsume condition ψ^n . Note that, if a service context different to \mathbf{v}_s is encountered prior to stabilising model ω^n , the observations encountered under condition ψ^n are considered as noise and ω^n is simply discarded.

4 Experiments and Results

We evaluate the performance of the proposed approach in terms of producing accurate quality value predictions in dynamic and user-dependent settings¹. We show the results (averaged over 100 runs) from the perspective of one service and one quality attribute q . An experiment run consists of a number of learning episodes of the service agent, each involving three steps: (1) *observing* value v_q for quality attribute q delivered by the service under user's context \mathbf{v}_u and service's context \mathbf{v}_s ; (2) *adjusting* the current configuration utilising this new observation $(v_q, \mathbf{v}_u, \mathbf{v}_s)$; and (3) *predicting* the expected quality value for the next user using the adjusted configuration. Further details are presented next.

4.1 Value Model Implementation

To implement the body M of each model $\omega \in \Omega$, we use the Naive Bayesian classifier [2]. In particular, given a quality attribute q and a corresponding observed user's context sample \mathbf{v}_u , the value v_q predicted for q is the one maximising the posterior probability $p(v_q|\mathbf{v}_u)$, given as $p(v_q|\mathbf{v}_u) = \frac{p(v_q) \times p(\mathbf{v}_u|v_q)}{p(\mathbf{v}_u)}$. Here: $p(v_q)$ is the prior probability of value v_q ; $p(\mathbf{v}_u)$ is the prior probability of sample \mathbf{v}_u (this is the same for all the values of q and thus could be omitted); and $p(\mathbf{v}_u|v_q)$ is the posterior probability of sample \mathbf{v}_u conditioned on value v_q . To simplify the computation cost of $p(\mathbf{v}_u|v_q)$, independence is usually assumed among the attributes of the sample, leading to: $p(\mathbf{v}_u, v_q) = \prod_{i=1}^m p(v_u^i|v_q)$. The estimation of probabilities $p(v_q)$ and $p(v_u^i|v_q)$ can be easily achieved via maintaining corresponding value counts, and thus the incremental learning function $learn_{q,u}$, corresponds to the update of these counts after each new service observation.

4.2 Dataset

We utilise a synthetic dataset inspired by STAGGER concepts [2], but is adapted to suit our problem. We assume: five possible outcomes for quality attribute q , $dom(q) = \{v_q^1, v_q^2, v_q^3, v_q^4, v_q^5\}$; three user-side context attributes, c_u^1 , c_u^2 , and c_u^3 , affecting q , each with three possible values, $dom(c_u^1) = \{v_u^{1,1}, v_u^{1,2}, v_u^{1,3}\}$, $dom(c_u^2) = \{v_u^{2,1}, v_u^{2,2}, v_u^{2,3}\}$, and $dom(c_u^3) = \{v_u^{3,1}, v_u^{3,2}, v_u^{3,3}\}$; one service-side context attribute, c_s , with three possible values, $dom(c_s) = \{v_s^1, v_s^2, v_s^3\}$; and three different service behaviours regarding attribute q , *behaviour 1* (associated

¹ Source code and data for the results presented in this paper are freely available from <http://jaspr.org/source-code>.

with v_s^1) where the actual value of q is v_q^1 (under $v_u^{1,1} \wedge v_u^{2,1}$) and is v_q^2 (otherwise), *behaviour 2* (associated with v_s^2) where the actual value of q is v_q^1 (under $v_u^{2,2} \vee v_u^{3,2}$) and is v_q^3 (otherwise), and *behaviour 3* (associated with v_s^3) where the actual value of q is v_q^4 (under $v_u^{1,2} \vee v_u^{1,3}$) and is v_q^5 (otherwise). The service switches from one behaviour to another at particular points, with such drifts being associated with changes in the value of service-side context attribute c_s .

4.3 Evaluation Strategies and Measure

We refer to the following quality value learning strategies. Strategy *MML*, our proposed multi-model learning approach. Strategy *SSL*, a simple summary-based learning approach, which predicts the quality value v_q with the highest prior probability, $p(v_q)$, based on all the observations so far and ignoring the user's context. Finally, strategy *SWL_w*, a sliding window based learning approach, a well known way in the literature of adapting to potential changes in incoming data [3]. It utilises the Naive Bayesian classifier presented in Sect. 4.1 as its main model, but maintains a fixed window of the latest w observations, based upon which the model is updated at each time step (this strategy accounts for the user's context, but ignores the service's context). By *SWL_{all}*, we refer to accounting for *all* the data observed so far. The performance of each strategy is evaluated by assessing its prediction accuracy at each time step, calculated as the success rate (i.e. $\frac{\text{number of successful predictions}}{\text{total number of predictions}}$) over the last 20 observations.

4.4 Results

To study the importance of user context awareness, we first assume a static service behaviour regarding attribute q (e.g. *behaviour 2*), and compare our learning strategy, *MML*, against the simple summary one, *SSL*. To simulate situations of imperfect user's contextual knowledge, attribute q is subjected to different levels of noise $\eta\%$. The results in Fig. 1(a) demonstrate that *MML* achieves an accuracy of over 80 % (at 0 % noise) after only 30 cycles, as opposed to *SSL* where the accuracy fluctuates around 50 % for the entire run. It is also evident that even with high noise levels (e.g. 30 % noise), *MML* still outperforms *SSL*, indicating the importance of contextual evidence even if imperfect.

To study adaptivity in dynamic environments, we now assume that the service follows the behaviour sequence 1-2-3-1-2-3, with each behaviour being fixed for 300 episodes. Figure 1(b) compares the adaptation strategies in the case of encountered *new behaviour* (i.e. changes during the first 900 episodes), with *stability* being set to 15. *MML* outperforms the other strategies, increasing the accuracy to over 90 % after just 30 observations from the change point. *SWL_{all}*, however, suffers from poor performance, especially after a change, where the learned model mostly reflects irrelevant observations. In fixed windowing strategies, increasing the window size results in a slower reactivity to a change since older irrelevant observations take longer to be forgotten, while smaller windows achieve faster adaptation but affect the prediction accuracy due to depending on insufficient number of observations. In the case of encountered

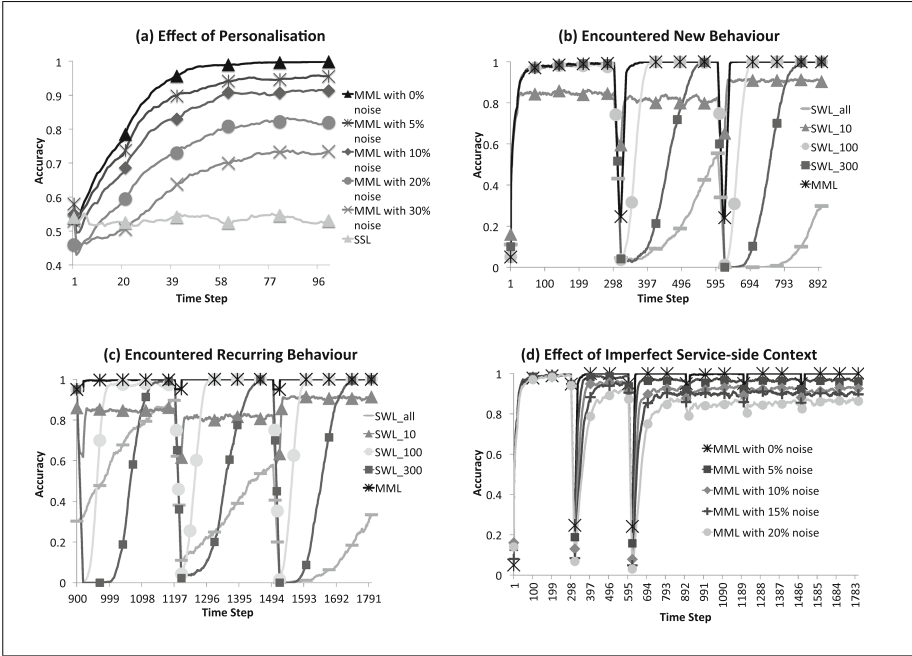


Fig. 1. Evaluation results

recurring behaviour (i.e. changes in the second 900 episodes), and unlike the other strategies, *MML* always maintains high accuracy (see Fig. 1(c)), eliminating the period of performance degradation after a change due to reusing an existing stable model. The effect of imperfect (e.g. incomplete) service-side context knowledge on *MML* is studied in Fig. 1(d), where context attribute c_s is subjected to various levels of noise $\eta\%$, and the results indicate robustness to noise.

5 Related Work

The QoS properties of services are important criteria upon which services are discovered, selected, and composed [4]. Accurate estimation of such properties has thus received much attention. Aschoff et al. [5] model the response time of a service as a random variable, with the exponentially weighted moving average being utilised for estimating the expected value of this variable at a particular time step according to historical data. Similarly, time series modelling based on ARIMA (AutoRegressive Integrated Moving Average) has been proposed by Amin et al. [6] for the purpose of QoS forecasting. Barakat et al. [7] provide probabilistic, multi-valued quality estimations for services via applying an online learning algorithm inspired by Policy Hill-Climbing based on past user ratings. Trust and reputation mechanisms have also been considered for the purpose of

accurate quality predictions [8], where an assessment of a QoS dimension's trustworthiness (e.g. a time-weighted average of the past service ratings) is undertaken prior to an interaction. In contrast to our approach, all such efforts rely on favouring recent observations to handle changes in the service's behaviour, without accounting for contextual clues, thus neglecting important evidence.

To facilitate personalised QoS information for users, a number of approaches utilise collaborative filtering for quality value prediction [9]. Although such approaches capture the user's context implicitly, they usually suffer from the data sparsity problem, unlike our approach, which explicitly exploits available user's contextual knowledge, which allows deriving *personalised* quality assessments for the user even in the absence of previous interactions with this user. Finally, like us, Lin et al. [10] explicitly incorporate the user's contextual attributes as input for the quality value prediction process. Yet, they do not account for changes in the service's behaviour associated with service-side context.

6 Conclusion

The paper presented a context-aware QoS learning approach for personalised and adaptive quality estimations. The learning is conducted via a service agent, which maintains a pool of quality prediction models; each characterising a particular service behaviour and providing personalised value predictions for users. Experimental results show that the approach achieves high accuracy and a faster change adaptation when compared to commonly adopted time-based learning.

Acknowledgments. This work was part funded by the UK Engineering and Physical Sciences Research Council as part of the Justified Assessments of Service Provider Reputation project, ref. EP/M012654/1 and EP/M012662/1.

References

1. Yang, Y., Wu, X., Zhu, X.: Mining in anticipation for concept change: proactive-reactive prediction in data streams. *Data Min. Knowl. Discov.* **13**, 261–289 (2006)
2. Widmer, G.: Tracking context changes through meta-learning. *Mach. Learn.* **27**, 259–286 (1997)
3. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. *ACM Comput. Surv.* **46**, 1–37 (2014)
4. Barakat, L., Miles, S., Poernomo, I., Luck, M.: Efficient multi-granularity service composition. In: *IEEE International Conference on Web Services*, pp. 227–234 (2011)
5. Aschoff, R., Zisman, A.: QoS-driven proactive adaptation of service composition. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) *Service Oriented Computing*. LNCS, vol. 7084, pp. 421–435. Springer, Heidelberg (2011)
6. Amin, A., Colman, A., Grunske, L.: An approach to forecasting QoS attributes of web services based on ARIMA and GARCH models. In: *IEEE International Conference on Web Services*, pp. 74–81 (2012)

7. Barakat, L., Mahmoud, S., Miles, S., Taweel, A., Luck, M.: An agent-based service marketplace for dynamic and unreliable settings. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) ICSOC 2014. LNCS, vol. 8831, pp. 169–183. Springer, Heidelberg (2014)
8. Maximilien, E. M., Singh, M. P.: Agent-based Trust Model Involving Multiple Qualities. In: 4th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 519–526 (2005)
9. Zheng, Z., Ma, H., Lyu, M.R., King, I.: QoS-aware web service recommendation by collaborative filtering. *IEEE Trans. Serv. Comput.* **4**, 140–152 (2011)
10. Lin, D., Shi, C., Ishida, T.: Dynamic service selection based on context-aware QoS. In: *IEEE International Conference on Services Computing*, pp. 641–648 (2012)

Service Composition (Short Papers)

Spatio-Temporal Composition of Crowdsourced Services

Azadeh Ghari Neiat^(✉), Athman Bouguettaya, and Timos Sellis

School of Computer Science and Information Technology,
RMIT, Melbourne, Australia
{azadeh.gharineiat,athman.bouguettaya,timos.sellis}@rmit.edu.au

Abstract. We propose a new composition approach for crowdsourced services based on dynamic features such as spatio-temporal aspects. The proposed approach is defined based on a formal crowdsourced service model that abstracts the functionality of crowdsourced data on the cloud in terms of spatio-temporal features. We present a new QoS-aware spatio-temporal union composition algorithm to efficiently select the optimal crowdsourced composition plan. Experimental results validate the performance of the proposed algorithm.

Keywords: Crowdsourced service · Spatio-temporal composition · Crowdsourced service composition · Spatio-temporal QoS

1 Introduction

The ubiquity of mobile devices enables to crowdsource sensor data. Storing, processing and managing continuous streams of crowdsourced sensed data pose key challenges [4]. Due to availability, low-cost and fast access to cloud services, the aggregation of crowdsourced sensor data on the cloud (i.e., crowdsourced Sensor-Cloud) provides a unique opportunity to address the above challenges.

We propose to harness the service paradigm as a key mechanism to turn crowdsourced sensor data into useful information. The service paradigm is a powerful abstraction hiding data-specific information which focuses on *how* data is to be used. In this regard, the functionality of crowdsourced sensor data on the cloud is abstracted as *crowdsourced* services which are easily accessible irrespective of the distribution of crowdsourced sensor data sources.

Because of the nature of crowdsourced sensors, mobility is an intrinsic part of the functional and non-functional aspects of crowdsourced services. This provides an opportunity to combine individual crowdsourced services to provide value-added services whenever they are available. However, this also presents challenges because of the hiding distributed, volatile and dynamic aspects including *spatio-temporal* dependencies. In this regard, we focus on spatio-temporal aspects as key parameters to query the crowdsourced Sensor-Cloud. The challenge can be more formally defined as finding the “best” spatio-temporal composition of crowdsourced services.

This paper focuses on providing a framework for spatio-temporal selection and composition of crowdsourced services. We first formally define a new spatio-temporal model for crowdsourced services and composition framework. In addition, we propose a spatio-temporal union composition algorithm. Our case study focuses on the use of WiFi hotspot sharing in a geographical area. Finally, a performance study of the proposed approach is presented.

The remainder of the paper is structured as follows: Sect. 2 formally defines the crowdsourced service model and composition framework. Section 3 illustrates the new QoS model. Section 4 details the proposed composition approach. Section 5 describes the evaluation of the approach. Section 6 concludes and highlights our future work.

Motivating Scenario. Let us assume that Sarah would like to find the best WiFi hotspot-covered path from ‘A’ to ‘B’. The process of finding the best journey is considered as a two-steps composition problem. In the first step as explained in [2], we consider a line segment as a service (e.g., a tram service) with a set of quality parameters. By applying STA* [3], we have a set of optimal travel plans from ‘A’ to ‘B’. In the second step, we take the first step output as an input and then look at WiFi coverages as a key parameter to determine the best travel plan. Our approach considers one path at a time for selecting the best coverage along each and every optimal plan. The novelty of this approach is considering QoS WiFi coverage as a service because of intrinsic complexity of WiFi coverage which usually includes area, signal strength and bandwidth. Therefore, we formulate the problem of selecting the best coverage as a composition of WiFi coverages on a journey from ‘A’ to ‘B’. Each basic WiFi coverage is offered by one hotspot provided by the crowd. Therefore, the entire coverage of a plan will be crowdsourced. We also assume that hotspots are *static*, i.e., the coverage does not change in time and space. Key to crowdsourcing hotspots are the spatio-temporal attributes which will be used for selecting and composing services. This paper focuses on the second level.

2 Spatio-Temporal Model for Crowdsourced Service

In this section, we propose a new formal spatio-temporal model for an atomic crowdsourced service and crowdsourced service composition framework.

2.1 Spatio-Temporal Model for Atomic Crowdsourced Services

We discuss the key concepts to model a crowdsourced service in terms of spatio-temporal features of crowdsourced sensor data.

Definition 1: Linear Composition Plan P . A linear composition plan P is modelled as a sequence of component line segment services [2] in the form of a trajectory $\{ \langle p_i, t_i \rangle, 1 \leq i \leq k \}$, where

- p_i is a geospatial coordinate set (x_i, y_i) ,
- t_i is a time instant.

Definition 2: Linear Plan Set \mathbb{P} . Given a source point ς and destination point ξ , a linear plan set \mathbb{P} is a set of all optimal linear composition plans from ς to ξ (Fig. 1(a)). We assume that \mathbb{P} is the output of applying a variation of STA* algorithm in our previous work [3] that return k optimal linear plans.

Definition 3: Sensor sen . A sensor sen is a tuple of $\langle sid, loc, sa, tsp \rangle$ where

- sid is a unique sensor ID,
- loc is the latest recorded location of sen ,
- sa is the specific sensing area centred at loc with the radius R_s ,
- tsp (timestamp) is the latest time in which sensor data related to a crowdsourced service is collected from sen .

Definition 4: Crowdsourced Service S . A crowdsourced service S is a tuple of $\langle id, SEN, space-time, F, Q \rangle$ where

- id is a unique service ID,
- $SEN = \{sen_i.sid | 1 \leq i \leq m\}$ represents a finite set of sensors sen_i collecting sensor data related to S . In this paper, we assume that each crowdsourced service consists of one sensor (i.e., $|SEN| = 1$),
- $space-time$ describes the spatio-temporal domain of S . In this paper, we restrict the $space$ of a service to a *surface area*. The $space$ is presented by a spatial square area A_s which is a minimum bounding square containing the sensing area of S (i.e., $S.SEN.sen_1.sa$). The $time$ is a tuple $\langle t_s, t_e \rangle$, where
 - t_s is a start-time of S ,
 - t_e is an end-time of S .

As can be seen in Fig. 1(b), the crowdsourced service is modelled as a Minimum Bounding Box MBB which is represented by (x_s, y_s, t_s) (i.e., bottom-left) and (x_e, y_e, t_e) (i.e., top-right),

- F describes a set of functions offered by S (e.g., providing WiFi hotspot),
- Q is a tuple $\langle q_1, q_2, \dots, q_n \rangle$, where each q_i denotes a QoS property of S .

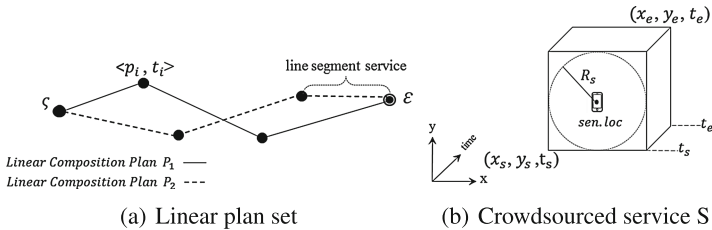


Fig. 1. Crowdsourced service model

2.2 Spatio-Temporal Model for Composite Crowdsourced Services

It is quite likely that a linear plan cannot be covered by a single crowdsourced service. In such cases, crowdsourced services may need to be composed to cover the linear plan. The following rule, called *spatio-temporal composability*, checks whether two component services are spatio-temporally composable.

- *Definition 5: Spatio-Temporal Composability.* Two component services S_k and S_l are spatio-temporally composable with respect to a linear plan P iff
 - $S_k \cap P \neq \emptyset$ & $S_l \cap P \neq \emptyset$ i.e., S_k and S_l intersect P .
 - S_l has overlap with the extended MBB of S_k (i.e., $MBB_{Ex}(S_k)$). The MBB_{Ex} (i.e., buffer area) is computed by extending each edge of the area of S_k by a distance d and also extending time edge by time period τ (Fig. 2(a)). The values of d and τ are assumed to be user-defined. For example, in our scenario d are τ are the maximum disconnection tolerant distance and time.
 - Two edge vectors $V_{p_s p_e}$ and $V_{loc_k loc_l}$ are in the same *direction*. The vectors $V_{p_s p_e}$ and $V_{loc_k loc_l}$ connect two vertices (p_s, p_e) and (loc_k, loc_l) respectively. p_s and p_e are the start-point and end-point of P and loc_k and loc_l are the sensed points of S_k and S_l , respectively (Fig. 2(a)). The vector direction is used as heuristic which is based on the premise that the best neighbours are going to be found in the direction where the traveler is going.
- As can be seen in Fig. 2(a), S_k and S_l are spatio-temporal composable. However, although S_m intersects P , it does not have overlap with $MBB_{Ex}(S_k)$. As a result, S_m and S_k are not spatio-temporal composable.

Given a plan P and a set of crowdsourced services $\{S_1, S_2, \dots, S_n\}$, we model a composite crowdsourced service as the total union area of component services that covers P (Fig. 2(b)). A composite crowdsourced service CCS is a sequence of component services $\{S_i, 1 \leq i \leq n\}$ where each pair of (S_i, S_{i+1}) is spatio-temporal composable. In the remainder of the paper, the service and composite service are used to refer to a crowdsourced service and composite crowdsourced service, respectively.

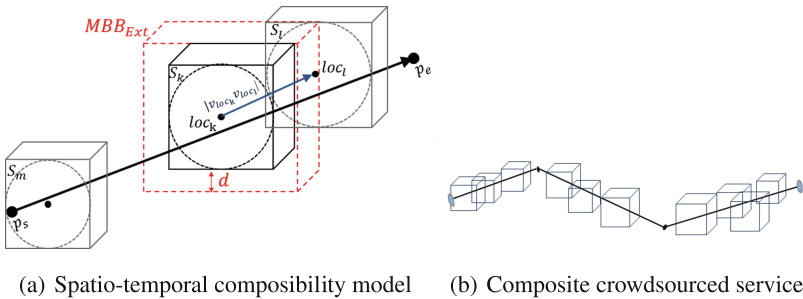


Fig. 2. Composite crowdsourced service model

2.3 Spatio-Temporal Index Data Structure for Crowdsourced Services

Indexing of services enables the fast discovery of services. We index services considering their spatio-temporal features using a 3D R-tree [5]. The 3D R-tree is a spatio-temporal index data structure which efficiently answers the range

queries of the type “report all objects located within a specific area during the given time interval”. The leaf nodes of the 3D R-tree represent actual services which are presented using *MBB* that encloses the area of a service (Fig. 3). To find composable services, called neighbours, we use *Spatio-TemporalSearch* algorithm [3] which searches through the 3D R-tree to find neighbours that intersect $MBB_{Ex}(S)$.

3 An Extensible Quality Model for Crowdsourced Service

Given the diversity of service offerings, an important challenge for users is to discover the ‘right’ service satisfying their requirements. We introduce novel QoS attributes for services. For the sake of clarity, we use a limited number of QoS attributes.

3.1 Quality Model for Atomic Crowdsourced Service

We propose to use spatio-temporal quality criteria which is part of describing the non-functional aspects of services:

- *Coverage (cov)*: Signal strength is associated with the coverage area of a service. The closer the user to the center of a service, the stronger WiFi signal is. We model $q_{cov}(S)$ using exponential attenuation probabilistic coverage model [1]. In this model, each service has a confident radius R_c (Fig. 4). The coverage $q_{cov}(S)$ varies from zero to one. Within the distance of R_c , the value of $q_{cov}(S)$ is 1 which means full signal. In the interval $(R_s - R_c)$, the value of $q_{cov}(S)$ approaches zero as the distance from the center increases. Beyond R_s , the value of $q_{cov}(S)$ is set to zero.
- *Capacity (cap)*: Capacity indicates the bandwidth for each user’s request which is important for uploading and downloading. Given an atomic service S , q_{cap} is computed as follows:

$$0 \leq \frac{DTR}{N_{cr}(S)} \leq 1 \tag{1}$$

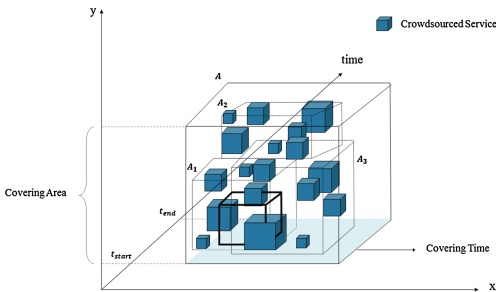


Fig. 3. Example of a 3D R-tree

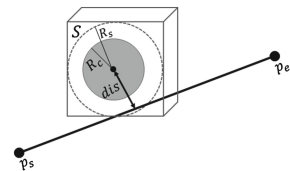


Fig. 4. Coverage QoS model

where DTR is the maximum speed at which the data can be transmitted for S (i.e., total available bandwidth) and $N_{cr}(S)$ is the number of concurrent requests that S can support. We assume that total available bandwidth is equally allocated.

3.2 Quality Model for Composite Crowdsourced Service

Aggregation functions are used to compute the QoS of composite services.

- *Coverage*: The coverage value of a composite service is the product of the coverage of all its component services. For each component service S_i , $q_{cov}(S_i)$ is computed with respect to the linear plan P as follows:

$$\begin{cases} 1 & 0 \leq \text{dis}(P, \text{loc}) \leq R_c \\ e^{-ka} & \text{dis}(P, \text{loc}) > R_c \end{cases} \quad (2)$$

where $a = \text{dis}(P, \text{loc}) - R_c$ and $\text{dis}(P, \text{loc})$ is the perpendicular distance from the sensed point of the component service loc to the linear plan P (Fig. 4). k is a sensor-technology dependent parameter which varies with the type of sensors and environment. The q_{cov} of the component service which is within the distance of R_c is 1. The q_{cov} value of the service lies within (R_c, R_s) exponentially decreases as the perpendicular distance increases. Since a component service intersects the linear plan, the value of dis is not beyond R_s (i.e., $q_{cov} \neq 0$).

- *Capacity*: The capacity value for a composite service is the average of the capacity of all its component services.

4 QoS-Aware Spatio-Temporal Union Composition Algorithm

We propose a new algorithm *FindBestPlan* to find the best linear plan of \mathbb{P} . The idea of our algorithm is to initially prune the search space with respect to \mathbb{P} and select a set of filtered services over the whole set of candidate services. We then divide the union composition into two phases: *local* for an individual component line segment service of the linear plan and *global* for the whole linear plan. The *local* phase computes the optimal union composition plan that covers the given line segment service. The *global* phase combines the optimal union composition plans of all its component line segment services obtained from the *local* phase. Finally, the best plan in \mathbb{P} is selected as the optimal solution. Algorithm 1 gives the details of *FindBestPlan* algorithm. In general, *FindBestPlan* algorithm works in the following four steps:

4.1 Crowdsourced Service Filtering

To improve the efficiency of the proposed approach, the first step is to reduce the search space of the algorithm. We develop a MBB that encloses a set of

Algorithm 1. FindBestPlan Algorithm**Input:** Linear plan set \mathbb{P} , 3D R-tree RT , maximum tolerant disconnection distance d , time period τ **Output:** The plan with the highest union utility-score in \mathbb{P} bestPlan

```

1: max-union-u-score=0
2: enclosingMBB = compute the enclosing MBB
3: Add all crowdsourced services inside the enclosing MBB of RT to a new 3D R-tree CRT
4: for each  $P \in \mathbb{P}$  do
5:   UCSList =  $\emptyset$  // The list of component services
6:   for each line segment service  $ls$  in  $P$  do
7:     UCSList.insert( UnionComposition (CRT,  $ls$ ,
8:                                      $d$ ,  $\tau$ )
9:   Compute union-u-score[P]
10:  if union-u-score[P] > max-u-score then
11:    max-u-score = union-u-score[P]
12:    bestPlan = P
13:  end if
14: end for
15: return bestPlan

```

services relevant to \mathbb{P} . The services outside this MBB are assumed to have little probability of being involved in the optimal composition plan. The enclosing MBB is represented by the lower-bound $[x_{min}, y_{min}, t_{min}]$ and upper-bound $[x_{max}, y_{max}, t_{max}]$, where x_{min} (resp. x_{max}) and y_{min} (resp. y_{max}) are the lowest (resp. highest) x-coordinate and y-coordinate among all coordinates of all optimal linear plans (Fig. 5). t_{min} and t_{max} are the minimum and maximum time value among all time instants of all optimal linear plans. 3D R-tree retrieves the services which are inside the enclosing MBB and have overlap with boundaries. All retrieved services are indexed by a new 3D R-tree (CRT) (2-3 in Algorithm 1).

4.2 Decomposition

The decomposition step divides each linear composition plan into elementary line segment services ls . Each ls is presented by a line segment of length 1 which consists of two consecutive tuples $\langle p_i, t_i \rangle, \langle p_{i+1}, t_{i+1} \rangle$ (Fig. 1(a)).

4.3 Local Union Composition

Given a line segment service ls , the local union composition step finds a composite service that covers ls . The spatio-temporal union composition problem can be modeled as a directed graph search problem in which each vertex has an associated space-time attribute of a service and each edge is associated with QoS attributes. If an edge exists, it means that there is a neighbour dependency between vertices. A virtual start-point vertex $(ls.p_s, ls.t_s)$ and virtual end-point vertex $(ls.p_e, ls.t_e)$ are added to the graph. The virtual vertices are connected to all neighbour services. Note that if no service is within the distance d of start-point and end-point, the search distance is increased until a service is found. The coverage and capacity of these neighbour services are set to neutral values of one and zero, respectively. We propose *UnionComposition* algorithm which is a variation of Dijkstra shortest path finding algorithm that minimizes the search cost function to find the optimal union composition plan from the start-point to end-point.

UnionComposition differs on *search cost* and *neighbour* functions. The *search cost* function of a union composition is defined as the following utility function [6]:

$$\text{union-u-score} = \sum_{Q_i \in \text{neg}} W_i \frac{Q_i^{\text{max}} - Q_i}{Q_i^{\text{max}} - Q_i^{\text{min}}} + \sum_{Q_i \in \text{pos}} W_i \frac{Q_i - Q_i^{\text{min}}}{Q_i^{\text{max}} - Q_i^{\text{min}}} \quad (3)$$

To find neighbour services (i.e. candidate services) of a service, we define a new *neighbour* function based on *Spatio-TemporalSearch* algorithm [3] which searches through the 3D R-tree to find neighbours. Then neighbours which intersect the line segment and are in the same direction of the segment are added to the candidate list.

The details of *UnionComposition* is shown in Algorithm 2. The algorithm starts finding neighbour services of the start-point of the line segment. The union-u-score of services in the candidate list are computed (4–6). The algorithm selects the candidate service with the lowest cost as the next candidate to be examined. Because the higher value of union-u-score shows the better QoS, we use (1 - union-u-score). The candidate service with the smallest union-u-score sets as the current service (L.8). For the current service, all of its unvisited neighbours are considered and their tentative u-scores are computed (14–18). If the current service is the neighbour of the destination (i.e., the search is successful) (9–11) or if the candidate list is empty (i.e., there is no composition plan) (L. 7), then stop. Otherwise, the algorithm selects the candidate service with the smallest tentative u-score and sets it as the new current service and continue (19–25).

Algorithm 2. UnionComposition Algorithm

<pre> Input: Line segment service ls, filtered 3D R-tree CRT, , maximum tolerant disconnection distance d, time period τ Output: Best crowdsourced composition plan CCS 1: compositionPlan = \emptyset // The plan of navigated ser- vices 2: visitedList = \emptyset // The list of services already evalu- ated. 3: candidateList = Spatio-TemporalSearch(CRT, $ls.p_s$, $ls.t_s, d, \tau$) \cap ls 4: for each $S \in$ candidateList do 5: compute union-u-score[S] 6: end for 7: while candidateList $\neq \emptyset$ do 8: currentS = A service in candidateList having the lowest (1-union-u-score) value 9: if currentS \in Spatio-TemporalSearch(CRT, $ls.p_e$, $ls.t_e, d, \tau$) then 10: return compositionPlan + currentS 11: end if </pre>	<pre> 12: visitedList.insert(currentS) 13: candidateList.remove(currentS) 14: NeighboursList = Spatio-TemporalSearch(CRT, currentS.loc, currentS.t_e, d, τ) \cap ls 15: for each $ns \in$ NeighboursList do 16: if $ns \notin$ visitedList and $ns.id \neq$ currentS.id then 17: tentative-u-score = union-u-score[ns] 18: end if 19: if $ns \notin$ candidateList or tentative-u-score \leq union-u-score[ns] then 20: compositionPlan[ns] = currentS 21: u-score[ns] = tentative-u-score 22: if $ns \notin$ candidateList then 23: candidateList.insert(ns) 24: end if 25: end if 26: end for 27: end while 28: output("No plan!") </pre>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

4.4 Global Union Composition

After performing the *local union* (L.7 Algorithm 1), each line segment service ends up with an optimal union composition plan that covers the line segment

service. The *global union* step takes the output of all *local unions*, combines them as a composite service and computes the union utility score of the composite service for each optimal linear plan (L.9 Algorithm 1). The best plan of \mathbb{P} is the plan with the highest union utility score of the composition process (10–13 Algorithm 1).

5 Experiments Results

In our experiment, we show the significance of our filtering step in terms of computation time. To the best of our knowledge, there is no usable and relevant real spatio-temporal service test case to evaluate our approach. Therefore, we focus on evaluating the proposed approach using synthetic services. Our evaluation sets a baseline upon which future work will be compared to. We run our experiments on a 3.60 GHZ Intel Xeon processor and 16 GB RAM under Windows 7. In our simulation, services are randomly distributed in a 70×70 region. The space and time attributes of services are randomly determined within the region range. The q_{cov} is assigned at runtime based on the distance between the service and the linear plan with respect to R_s, R_c and k parameters. To obtain a more realistic approach, we use heterogeneous sensors for services by varying the values of these parameters which are set as follows: $k = 0.5$ and $R_s \in [4, 6]$ and $R_c \in [1, 3]$ are randomly selected. The $q_{cap} \in [50, 100]$ is also randomly generated. We set the weights as $W_{cov} = 0.5$ and $W_{cap} = 0.5$. The remaining service parameters are also randomly generated using a uniform distribution. All experiments are conducted 100 times and the average results are computed. Each experiment starts from a different source and destination point which are randomly generated. For each experiment we apply a variation of STA* [3] and select top two optimal linear plans. *FindtBestPlan* algorithm is then executed to find the best plan based on union composition cost.

We study the significance of our filtering stage in terms of the computation time. We define the optimality ratio as follows:

$$Optimality\ ratio = \frac{ct_{wf} - ct_f}{ct_{wf}} \tag{4}$$

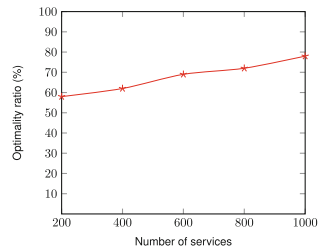
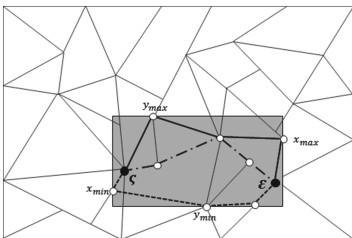


Fig. 5. Example of an enclosing MBB **Fig. 6.** Optimality in terms of computation time

where ct_{wf} is the execution time of our algorithm *FindtBestPlan* without filtering and ct_f is the execution time of *FindtBestPlan* by applying filtering stage. We measure optimality ratio while fixing the default map size and varying the number of services from 200 to 1000 with an iteration range of 200. Figure 6 illustrates that filtering phase produces a satisfying optimality (i.e., more than 58%). It means that applying filtering stage significantly reduces the computation time which confirms our expectation about its impact on the computation time. The results also show that the optimality ratio increases slightly along with the number of services.

6 Conclusion

We introduce a new spatio-temporal union composition algorithm to efficiently select the optimal union composition plan considering multiple new QoS criteria. We demonstrate that our algorithm has a satisfying efficiency in terms of optimality. Future work focuses on moving and transient crowdsourced services.

Acknowledgments. This research was made possible by NPRP 7-481-1-088 grant from the Qatar National Research Fund (a member of The Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

1. Altinel, İ.K., Aras, N., Güneş, E., Ersoy, C.: Binary integer programming formulation and heuristics for differentiated coverage in heterogeneous sensor networks. *Comput. Netw.* **52**(12), 2419–2431 (2008)
2. Neiat, A.G., Bouguettaya, A., Sellis, T., Dong, H.: Failure-proof spatio-temporal composition of sensor cloud services. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) *ICSOC 2014*. LNCS, vol. 8831, pp. 368–377. Springer, Heidelberg (2014)
3. Neiat, A.G., Bouguettaya, A., Sellis, T., Ye, Z.: Spatio-temporal composition of sensor cloud services. In: *2014 IEEE International Conference on Web Services (ICWS)*, pp. 241–248. IEEE (2014)
4. Hossain, M.A.: A survey on sensor-cloud: architecture, applications, and approaches. *Int. J. Distrib. Sens. Netw.* **2013**, 1–18 (2013)
5. Theoderidis, Y., Vazirgiannis, M., Sellis, T.: Spatio-temporal indexing for large multimedia applications. In: *Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems*, pp. 441–448. IEEE (1996)
6. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. *IEEE Trans. Softw. Eng.* **30**(5), 311–327 (2004)

Design for Adaptation of Distributed Service-Based Systems

Antonio Bucchiarone, Martina De Sanctis, Annapaola Marconi,
Marco Pistore, and Paolo Traverso^(✉)

Fondazione Bruno Kessler, Via Sommarive, 18, Trento, Italy
{bucchiarone,msanctis,marconi,pistore,traverso}@fbk.eu

Abstract. Internet of Services applications need to cope with a continuously changing environment, both in terms of the context in which they operate, and of the services, users and providers involved. In this setting, adaptivity is to be considered an intrinsic characteristic of applications rather than an exception to be handled. In this paper we propose a design for adaptation approach that fully exploits the advantages of the service-oriented paradigm to support the development and operation of service-based applications operating in highly dynamic environments. The approach is based on dynamic and incremental service composition and re-configuration techniques and is evaluated on a real-world scenario in the Smart Cities domain.

1 Introduction

The Internet of Services (IoS) foresees a future Internet in which the provisioning of, access to and use of services will be as widespread as content is today. The urgent need for a more efficient and sustainable society, together with the spread of ubiquitous communication networks, highly distributed wireless sensor technology, and intelligent management systems, makes the *Smart City ecosystem an ideal ground for IoS*. In this setting, the role of service-oriented computing is to enable the integration and interplay between new and legacy city services to support the creation and delivery of innovative and efficient services for the citizens [10].

A key challenge that needs to be overcome for this to become a reality, is the capability of dealing with the *continuously changing complex environment* in which Smart City applications operate. Consider for instance the case of a “smart children’s mobility system”, supporting service users (parents, children) and providers (drivers, teachers, traffic aids, volunteers) in their daily operation of children mobility services (e.g., school buses, walking buses, bike trains, ride-sharing among parents). The implementation of such a system requires to deal with a variety of heterogeneous services provided by autonomous entities (e.g., registration services provided by the school, volunteer management, safety and traffic information from local police system, access to smart and wearable devices). In this setting, changes are not only frequent, they are an inner characteristic of the system. In particular, the system should be resilient to changes

in existing services and in user requirements, as well as be open and extensible for new functionalities and facilities to become part of the system. This is made particularly challenging by the high degree of connection and interdependencies among system components which are provided by autonomous entities [1]. There is therefore the need to develop applications that are *adaptable* “by design” and to provide a *dynamic and scalable runtime environment* that makes them resilient to the aforementioned changes.

In this paper we propose a *design for adaptation approach* that supports the development, deployment and execution of service-based systems operating in dynamic environments and we discuss its effectiveness when applied to Smart cities applications like the “smart children’s mobility system”. The approach exploits advanced techniques for dynamic and incremental service composition [2], to effectively deal with changes occurring at different levels in the system. Each system component is modeled in terms of the functionalities it provides and of the conditions under which each functionality can be used (e.g. context/situation-related properties). At the same time, it is possible to partially specify the implementation logic of each component that is then automatically refined with (one or a composition of) functionalities provided by other components. This results in a dynamic network of components, where most changes can be handled at a local level, that is, affecting only those components directly related to the change and making the change transparent to the rest of the system.

The rest of the paper is structured as follows. Section 2 introduces the children mobility scenario used as a reference throughout the paper; Sect. 3 presents in details the proposed approach for the definition and operation of service-based systems that are adaptable by design; finally, Sect. 4 discusses the effectiveness and performances of the approach while Sect. 5 presents some related works and conclusions.

2 Motivating Scenario

Independent and active mobility has been proven to be fundamental for the development of children and adolescents. Plenty of initiatives have been set up in our cities supporting children’s freedom of movement. Just to cite a few examples, we have school buses, walking buses (children walking together to school supervised by volunteer parents), children bike-trains (similar to walking buses, by bike), volunteers at crosswalks, ride-sharing among parents. However, if the aim is to deliver “smart children mobility services”, these services cannot be managed each by itself, but they should become part of an integrated mobility solution, the *Smart Children Mobility System* (SCMS), that supports service users (parents, children) and providers (teachers, traffic aids and volunteers) in their daily operation and management of the different mobility services.

Figure 1 presents an overview of the different components in the SCMS. If we consider the part of the system related to the walking school bus (WSB), we notice that the system should support the registration of children, parents and

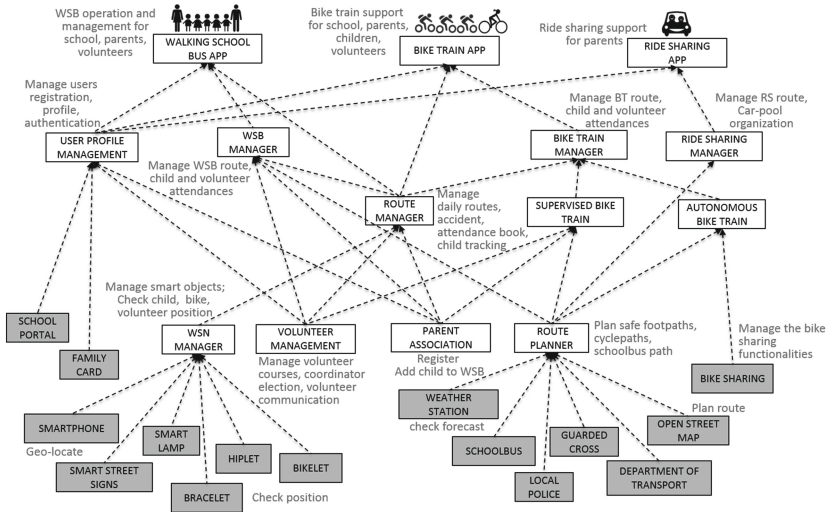


Fig. 1. Smart children mobility system: a partial overview of the system

volunteers and their access to the system (**User Profile Management** component), the training of volunteers (**Volunteer Management** component), the organization of routes (**WSB Manager** component) taking into account the needs of families but also route safety (e.g., presence of sidewalks, traffic situation, guarded crosswalks). The daily operation of the service requires to handle children attendance and volunteers availability for each route (**Route Manager**), the compilation of attendance books, tracking of children and volunteer position (**WSN Manager**), as well as managing possible exceptions (e.g. find a substitute for a volunteer, change the route due to roadworks, suspend the route due to weather conditions). Some components are in common with the bike train and ride sharing (e.g. **Route Manager**, **Volunteer Management**, **User Profile Management**). This allows not only replication avoidance, but also enabling synergies among the different services (e.g., exploit the ride-sharing service to cover a WSB route in case of bad weather).

A key characteristic of the system is the *variety and heterogeneity of services* involved: from domain-specific functionalities (e.g., management of walking bus routes, compilation of attendance books) to general-purpose ones (e.g., access management, user tracking); from back-end functionalities (in gray in the Figure) requiring the interaction with third party systems and devices (e.g. retrieving cycle lines, traffic/safety street information, interacting with smart objects) to front-end ones (e.g., Apps to be accessed by parents, volunteers, teachers). Moreover, the SCMS needs to deal with the *dynamicity* of the scenario, both in terms of the variability of services involved and of context changes affecting its operation. In particular, the system should be *open and extensible*, which means that new services (e.g., a new tracking device), as well as changes in existing

services (e.g., changes in parents authorization procedure, changes in any third party system) should require minimum maintenance. This is made particularly challenging by the collective nature of the services to be provisioned, since their operation requires the collaboration of different autonomous actors (school, parents associations, transport departments, local police), and it results in a *high degree of connection and interdependencies* among system components (as shown in Fig. 1).

3 General Framework and Approach

In this section we present the proposed approach for modeling and executing distributed adaptive service-based systems, such as the *SCMS* described in Sect. 2.

The system is modeled through a set of *domain objects* representing system components. Each domain object is characterized by a *core* process, implementing its own behavior, and a set of process *fragments*, representing the functionalities it provides (see Fig. 2). Fragments [5, 11] are executable processes that communicate with the core process and that can be received and executed by other domain objects.

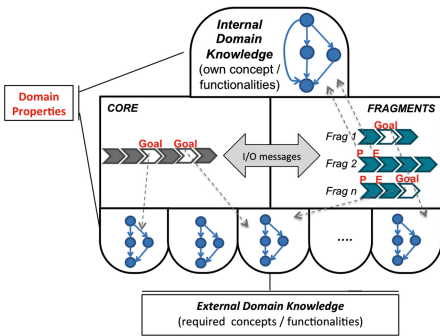


Fig. 2. Domain object model

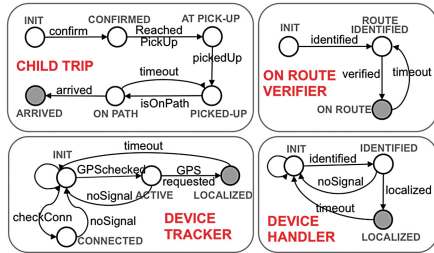


Fig. 3. Domain properties modeled as STS

Unlike traditional system specification, where components’ behavior are completely specified, our approach allows the partial specification of the expected operation of domain objects through *abstract activities* that can be refined at run-time according to the fragments offered by the other domain objects in the system. For instance, the *Device manager* (see Fig. 4) can partially define the functionality for the localization of a device. Different smart devices can join the system and publish different tracking procedures. At run-time, when the device to be tracked is known (e.g. smart bracelet), the *Device manager* will use the fragments offered by the specific device to refine its abstract activity and to eventually locate the device. Abstract activities can be used both in the core process of a domain object as well as in the fragments it provides. The latter

case is more complex, and enables a higher level of dynamicity, since it allows a domain object to expose a partially specified fragment whose execution does not rely only on communications with its core process but also on fragments provided by other domain objects, thus enabling a “chain of refinements”.

These dynamic features offered by the framework rely on a set of concepts, describing the operational environment, on which each domain object has a partial view. In particular (see Fig. 2), the *internal domain knowledge* captures the behavior of the domain concept implemented by the domain object, while the *external domain knowledge* represents domain concepts that are required to accomplish its behavior but for whose implementation it relies on other domain objects. Domain knowledge (both internal and external) is defined through *domain properties*, each giving a high-level representation of a domain concept (e.g. WSB route, child trip, device handler).

Consider for instance the domain property **Child trip** in Fig. 3 that models the typical daily trip of a child using a mobility service offered by the SCMS (e.g. walking bus). The participation of the child needs to be **CONFIRMED**, than the child reaches the pick up point (**AT PICK-UP POINT**) and she is **PICKED-UP** by a volunteer when she joins the ride. During the journey she might be **ON-PATH** (this is used to model the fact that at some times the system is certain of her position) and she eventually reaches the school (**ARRIVED**).

Each abstract activity is defined in terms of the *goal* it needs to achieve, expressed as domain properties’ states to be reached. It is automatically refined at run-time, considering the set of fragments currently provided by other domain objects, the current domain knowledge configuration, and the goal to be reached. Activities in processes and fragments are annotated with *preconditions* and *effects*. Preconditions constrain the activity execution to specific domain knowledge configurations. For instance, in Fig. 4, the precondition **P2: Device tracker = active** says that, to execute the fragment **Geo-locate device** in the **Bracelet** domain object, the domain property **Device tracker** (see Fig. 3) must be in the state **active**. Effects model the expected impact of the activity execution on the domain and represent its evolution in terms of domain properties events. For instance, in Fig. 4, the effect **E2: On route verifier.verified** models the evolution of the **On route verifier** domain property (see Fig. 3).

Preconditions and effects are used to model how the execution of fragments is constrained by and evolves the domain knowledge. This information is used to identify the fragment (or composition of fragments) that can be used to refine an abstract activity in a specific domain knowledge configuration.

The resulting adaptive system is a dynamic network of domain objects. Potential dependencies (*soft dependencies*, from here on) are established between a domain object and all those domain objects in the system whose provided functionality (internal domain knowledge) matches with one of its required behaviors (domain property in its external domain knowledge). A soft dependency between two domain objects becomes a *strong dependency* if, during the system execution, they inter-operate by injecting and executing a fragment. The external domain knowledge of a domain object is not static since, if during a refinement

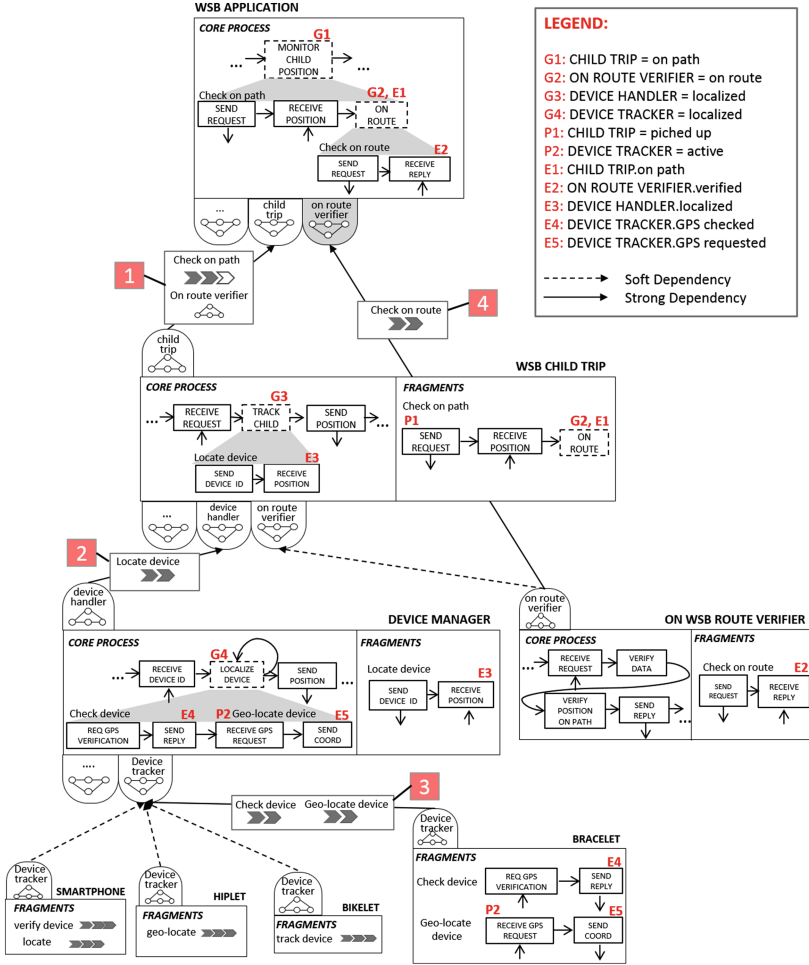


Fig. 4. A detailed example of dynamic refinements on the SCMS scenario

a domain object injects a fragment containing abstract activities in its own core process, it receives also the domain properties on which the fragment execution relies on, thus spanning its external knowledge. This dynamicy is reflected in the soft dependencies between domain objects because new dependencies might be established due to refinements.

In the following we present in details the different forms of refinement supported by the approach and how the adaptive system evolves at run-time. Consider, for instance, the scenario in Fig. 4 where a parent, or the school, wants to verify if a child is on the route of the WSB to which she has been subscribed.

Step 1. During the execution of the core WSB application process the abstract activity monitor child position needs to be refined. Its goal G1: Child Trip =

on path is defined over the `Child Trip` domain property (see Fig. 3). The refinement mechanism is triggered. The `WSB child trip` domain object implements the `child trip` domain property and exposes, among others, the `fragment check on path`. Supposing that this fragment is selected for the refinement, the first step of the refinement process consists in the injection of the `check on path` fragment in the behavior of the `WSB application`. Moreover, since the injected fragment contains an abstract activity, namely `on route`, the `WSB application`, together with the fragment, inherits the domain property on which the goal `G2: On Route Verifier = on route` is defined (Step 1 in Fig. 4). Thus, the `WSB application` domain object dynamically extends its knowledge boundaries and will use the received domain property to discover new domain objects (i.e., `on WSB route verifier`) and their related fragments. This step shows how fragments, together with domain knowledge, are exchanged by domain objects and injected at runtime in the core process. In particular, the `WSB child trip` domain object does not refine the `on route` abstract activity in its fragment, thus leaving the responsibility to refine it to the receiver domain objects.

Step 2. The `check on path` fragment, executed in the `WSB application` core process, communicates with the internal process of `WSB child trip`, which implements the functionality. During the execution of the `WSB child trip` process, the abstract activity `track child` needs to be executed. This activity has been defined as abstract since its execution depends on the tracking procedure supported by the device. The goal `G3: Device Handler = localized on track child` activity is expressed over `Device handler` domain property (see Fig. 3). The `device handler` behavior is offered by the `Device manager` through its fragment `Locate device`. A strong dependency is established between `WSB child trip` and `Device manager`, and the `locate device` fragment is injected in place of the `track child` activity (Step 2 in Fig. 4). This is an example in which, as opposite to Step 1, a domain object has an abstract activity in its core behavior, thus hiding to potential users of his fragments the fact that his behavior depends on third parties fragments.

Step 3. The execution of `Locate device` starts the execution of the `Device manager`'s core process, which allows a device to be localized. The `Device manager` process contains an abstract activity, `Localize device`, which can be refined in different ways depending on the device to be tracked. This allows the `Device manager` to track any kind of smart object connected to the network, provided that it offers the possibility to be tracked. In our example, supposing that the child to be tracked has a bracelet, the `Localize device` activity is refined exploiting the functionalities offered by the `Bracelet` domain object. In this case, it offers the `Check device` fragment to verify the GPS connection's availability and, the `Geo-locate device` fragment to ask for the current position. Since the `Geo-locate device` fragment, needed to accomplish the goal `G4`, is constrained by the precondition `P2: Device tracker = active`, the `Check device` fragment is selected and composed with it as its execution, thanks to effect `E4`, satisfies the precondition `P2`. The obtained fragments' composition is then injected in place of the `localize device` activity (Step 3 in Fig. 4).

Step 4. The execution goes back to **Check on path** fragment in the core process of **WSB application**. Once child position has been obtained, last step consists in verifying if the child is on the route of the WSB. The **On route** abstract activity can be refined. The **WSB application** exploits the domain knowledge received by the **WSB Child Trip** in Step1, to find the fragments to refine the abstract activity. The **Check on route** fragment is selected and injected in the **WSB application** (Step 4 in Fig. 4) and the whole refinement process ends.

4 Evaluation

In this Section we evaluate the proposed solution both in terms of its effectiveness in defining adaptive systems that are resilient to a wide range of changes and in terms of the scalability of the automated composition techniques used for the refinement of abstract activities.

TYPE	CHANGE	PROBABILITY	ADAPTIVE SYSTEM MODEL		RUNNING INSTANCES	
			IMPACT	SUPPORTED SOLUTION	IMPACT	POSSIBLE SOLUTIONS
SYSTEM	Add component (internal/external)	High	Local	Add domain object, new soft dependencies	None	-
	Add provided functionality	High	Local	Add fragment in domain object, new soft dependencies	None	-
	Behavioral / structural change in provided functionality	Medium	Local	Change fragment specification (activities, preconditions, effects)	None / Local	Multiple process variants (None). Re-refinement (Local).
	Remove functionality	Low	Local	Remove fragment, reduction in soft dependencies	None -> Broad	Multiple process variants (None). Re-refinement chain (Broad).
	Remove component	Low	Local	Remove domain object, reduction in soft dependencies	None -> Broad	Multiple process variants (None). Re-refinement chain (Broad).
DOMAIN	New domain concept	Medium	Local	Add domain property and new domain objects/frames implementing it	None	-
	Change in a domain concept	Low	Broad	Update domain property and all related fragment annotations in domain objects	None -> Broad	Multiple process variants (None). Multiple re-refinement chains (Broad).

Fig. 5. Impact of system- and domain-level changes on an adaptive system instance

In Fig. 5 we list a set of changes typical of systems operating in the Internet of Services. For each change, together with its *probability* to occur, we present its impact on the adaptive system. We distinguish *system-level* and *domain-level* changes. Among system-level changes we have: the need to include new components (e.g. a new kind of wearable device in the SCMS), remove components, add/remove functionalities to existing components (e.g., the new version of the WSB Child Trip component allows changing pick-up point for a specific ride), as well as changing the operation (both in terms of behavior or data structures) of existing functionalities. With domain-level changes we mean changes in the domain concepts on which the dynamic operation of the adaptive system relies on (domain knowledge model).

When analyzing the resilience of the system to changes, we consider both the impact on the adaptive system model, that is, domain objects models on which future instances will be based, and on the running instances. In both cases we

present the impact of the change in terms of its scope (*local* to the affected component vs *broad*) and of the activities to be performed to apply the change to the system (*Solution*).

As emerges from Fig. 5, the resulting adaptive system model is resilient to all system-level changes (local impact). Moreover, the only change at domain-level having a broad impact is the case in which a domain concept needs to be revised. However, this latter change is the least likely to occur, since it implies a revision in the standard operation of a domain concept. This is usually related to changes in a business procedure or regulation, or when the way of doing something changes in a disruptive way (e.g., a completely new way of tracking a device).

If we consider the impact on running instances the situation is more complex. This is due to the fact that process instances, within domain objects instances, might be connected through strong dependencies, since some abstract activities have been refined with process fragments. For this reason, if the change concerns adding a new component or functionality, or a new domain concept to the system, then there is no impact at all on the running instances; whereas in some other cases the impact might span the instances subject of change and affect other instances in the system (broad impact). In Fig. 5 we suggest some *possible solutions* to limit the impact of the change or, in case this is not possible, to deal with it without affecting the system operation. A very effective solution, that makes every change transparent to running instances, is based on the combination of the proposed approach with techniques supporting multiple process variants (e.g. approaches based on software product lines such as the one in [9]). This would allow the definition of a new variant of the system according to the required change, on which all new instances will be based, and keep the old version of the system as a sibling variant on which running instances can continue their operation.

To conclude, we will briefly discuss the scalability of the automatic refinement techniques on which the approach is based. The key question is whether it is feasible to solve large number of refinement problems at run-time, each involving a potentially large set of domain objects and available fragments. The refinement mechanisms proposed in Sect. 3 have been implemented in the ASTRO-CAptEvo Framework [2] and tested on a real-world car logistic scenario (for a detailed description of the experiments please refer to [2]). The experiments consider over 2500 refinement problems on an adaptive system with 29 domain object types, 69 process fragment models and 40 domain properties models. On average, an adaptation problem contained 7 process fragments that could potentially be used in the final solution, and, as such, were presented in the planning domain. The experiments show that 90 % of problems were resolved within 0.1 seconds and 99 % within 10 seconds. These results, based on advanced optimization (problem reduction) and solution reuse techniques, clearly prove the feasibility of on-the-fly refinement for adaptive systems.

5 Related Works and Conclusions

In this paper we propose a design for adaptation approach that, exploiting advanced service refinement and re-configuration techniques, supports the design, development, and operation of service-based systems that are resilient to a wide range of changes. Various other approaches have been proposed in this direction. In the following we consider those approaches that might be applied in scenarios similar to the SCMS presented in this paper.

We will start by analyzing *rule-based approaches*. We mention MoDAR [12], a model-driven approach for the development of adaptive service-based systems. Rules are used to capture the variable part of a business process and linked in specific cutting points of the base process. In [8], the authors tackled the problem of the unpredictable execution of service-based applications by modeling composite services as artifacts that can change at runtime. Here, rules are used to model adaptation needs (events) and adaptation actions, from the design-time phase. However, rules are not suitable for managing continuous and unpredictable changes in *open* environments, since they require human intervention to be revised.

Another category of approaches, quite close to our proposal, that emerged in recent years are *artifact-centric approaches*. In [7] the authors present a formal framework defining *Business Artifacts* which represent conceptual entities made of their attributes and states, their tasks modeling services performed on such artifacts and business rules defined in ECA style specifying the life-cycle of an individual artifact, as well as the control logic of a process executing between interacting artifacts. Although the work in [7] supports flexibility and reusability, it suffers from the same limitations of rule-based approaches described in previous paragraph.

Other works [3,9] exploit the concepts of dynamic software product lines (DSPL) [6]. In DSPL a software family is analysed as a whole and both common and reusable assets are established, together with the possible customizations of the application. Feature models are used to specify alternative variations that can be used for adaptation. In [3] the authors bring forward the idea of their previous approach, called DAMASCo [4], which mainly consists in allowing services reuse in pervasive systems, and provide an extension based on both SOA and DSPL. Feature models are used to represent the variability of the services by modeling families of adaptable software products and to allow, as a consequence, the realization of dynamic service composition in a context-aware manner. In [9] the authors present LateVa, where similar processes containing common and variable parts are called process variants. Base process models are annotated with variation points where fragments can be dynamically entered. In this way, both software reuse and run-time variability are addressed. The main limitation of DSPL-based approaches is that they assume a close world where process variants (and thus fragments) are pre-defined. For this reason, these approaches do not fit well in open environments in which system components, and their provided functionalities, can enter or leave the system in any moment.

Although the approach proposed in this paper might overcome some limitations of existing works in the field, there are several open issues we would like

to deal with in the near future. Among them, the most important is an on-the-field evaluation of the approach experimenting the SCMS with end-users. A key extension is the possibility of verifying the current state of the external domain knowledge of each domain object through monitoring facilities offered by other domain objects (at the moment the state evolves considering only the effect annotations on the received fragments and might not be aligned with the real world situation). Another important extension concerns the support for other forms of run-time adaptation (e.g., reaction to a context change observed by monitoring the environment).

Acknowledgment. This work is partially funded by 7th Framework EU-FET project 600792 ALLOW Ensembles.

References

1. Bucchiarone, A., Cappiello, C., Di Nitto, E., Kazhamiak, R., Mazza, V., Pistore, M.: Design for adaptation of service-based applications: main issues and requirements. In: Dan, A., Gittler, F., Toumani, F. (eds.) *ICSOC/ServiceWave 2009*. LNCS, vol. 6275, pp. 467–476. Springer, Heidelberg (2010)
2. Bucchiarone, A., Marconi, A., Mezzina, C.A., Pistore, M., Raik, H.: On-the-Fly adaptation of dynamic service-based systems: incrementality, reduction and reuse. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) *ICSOC 2013*. LNCS, vol. 8274, pp. 146–161. Springer, Heidelberg (2013)
3. Cubo, J., Gamez, N., Fuentes, L., Pimentel, E.: Composition and self-adaptation of service-based systems with feature models. In: Favaro, J., Morisio, M. (eds.) *ICSOC 2013*. LNCS, vol. 7925, pp. 326–342. Springer, Heidelberg (2013)
4. Cubo, J., Pimentel, E.: DAMASCo: a framework for the automatic composition of component-based and service-oriented architectures. In: Crnkovic, I., Gruhn, V., Book, M. (eds.) *ECSA 2011*. LNCS, vol. 6903, pp. 388–404. Springer, Heidelberg (2011)
5. Eberle, H., Unger, T., Leymann, F.: Process fragments. In: Meersman, R., Dillon, T., Herrero, P. (eds.) *OTM 2009, Part I*. LNCS, vol. 5870, pp. 398–405. Springer, Heidelberg (2009)
6. Hallsteinsen, S.O., Hinchey, M., Park, S., Schmid, K.: Dynamic software product lines. *IEEE Comput.* **41**(4), 93–95 (2008)
7. Hull, R., Damaggio, E., De Masellis, R., Fournier, F., Gupta, M., Heath, F.T., Hobson, S., Linehan, M.H., Maradugu, S., Nigam, A., Sukaviriya, P.N., Vaculín, R.: Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events. In: *DEBS 2011*
8. Hussein, M., Han, J., Yu, Y., Colman, A.: Enabling runtime evolution of context-aware adaptive services (2013)
9. Murguzur, A., Trujillo, S., Truong, H.L., Dustdar, S., Ortiz, Ó., Sagardui, G.: Run-time variability for context-aware smart workflows. *IEEE Software* **32**, 52–60 (2015)
10. Pistore, M., Traverso, P., Paolucci, M., Wagner, M.: From software services to a future internet of services. In: *Future Internet, Assembly*, pp. 183–192 (2009)
11. Sirbu, A., Marconi, A., Pistore, M., Eberle, H., Leymann, F., Unger, T.: Dynamic composition of pervasive process fragments. In: *ICWS*, pp. 73–80. IEEE (2011)
12. Yu, J., Sheng, Q.Z., Swee, J.K.Y.: Model-driven development of adaptive service-based systems with aspects and rules. In: Chen, L., Triantafillou, P., Suel, T. (eds.) *WISE 2010*. LNCS, vol. 6488, pp. 548–563. Springer, Heidelberg (2010)

Industry Track Papers

Automatic Deployment of Services in the Cloud with Aeolus Blender

Roberto Di Cosmo¹, Antoine Eiche², Jacopo Mauro³(✉), Stefano Zacchiroli¹,
Gianluigi Zavattaro³, and Jakub Zwolakowski¹

¹ University of Paris Diderot, Sorbonne Paris Cité, PPS, CNRS, Paris, France

² Mandriva S.A, Paris, France

³ Department of Computer Science and Engineering,
University of Bologna/INRIA FoCUS, Bologna, Italy
jmauro@cs.unibo.it

Abstract. We present *Aeolus Blender* (Blender in the following), a software product for the automatic deployment and configuration of complex service-based, distributed software systems in the “cloud”. By relying on a configuration optimiser and a deployment planner, **Blender** fully automates the deployment of real-life applications on OpenStack cloud deployments, by exploiting a knowledge base of software services provided by the Mandriva Armonic tool suite. The final deployment is guaranteed to satisfy not only user requirements and relevant software dependencies, but also to be optimal with respect to the number of used virtual machines.

1 Introduction

The cloud market is now a reality able to modify companies behaviour. The needs for solutions or efficient tools to support development activities for the profitability of the company is becoming more and more important. The new perspectives of IT usage (mobility, social networks, Web 2.0, Big Data) has brought the world into a new digital revolution. The first consequence is an explosion in the needs for computing and storage. According to an IDC study [23], digital data created in the world will increase to 40 Zettabytes in 2020. Faced with this, CIOs need to change, becoming more and more service-based and evolve their IT towards virtualized platforms and cloud (IAAS, PAAS, and SAAS) to address issues related to infrastructure growth, the need for power and provision computing resources on demand from internal or third party.

The CloudIndex study [34], conducted in partnership with Capgemini and Orange Business Services, indicates that 30% of respondents (300 companies)

This work was supported by the EU projects FP7-610582 *Envisage: Engineering Virtualized Services* (<http://www.envisage-project.eu>), FP7-644298 *HyVar: Scalable Hybrid Variability for Distributed, Evolving Software Systems* (<http://www.hyvar-project.eu>), and the ANR project ANR-2010-SEGI-013-01 Aeolus. It was partially performed at IRILL, center for Free Software Research and Innovation in Paris, France, <http://www.irill.org>.

have a cloud strategy and three quarters of them are planning to dedicate resources to this strategy. Two main directions emerge: homogenise the application portfolio and facilitate the deployment of applications.

Driven by this business need, several tools have been developed and used routinely to help system architects and administrators to automate at least some of the deployment and configuration phases of the complex service application in the cloud. For instance, configuration managers like Puppet [35] and Chef [33] are largely used by the “DevOps” community [13] to automate the configuration of package-based applications. Domain specific languages like ConfSolve [22] or Zephyrus [10] can be used to compute—starting from a high-level partial description of the application to be realised—an (optimal) allocation of the needed software components to computing resources. Tools like Engage [15] or Metis [26] synthesise the precise order in which low-level deployment actions should be executed to realise the desired application.

Despite the availability of such tools, the mainstream approach for deploying cloud applications is still to exploit pre-configured virtual machines images, which contain all the needed software packages and services, and that just need to be run on the target cloud system (e.g., Bento Boxes [16], Cloud Blueprints [8], and AWS CloudFormation [2]). However, the choices of the services to use (e.g., WordPress installed with Apache or Nginx, with NFS or GlusterFS support) lead to an explosion of configurations that can hardly be matched by the offered set of pre-configured images. Moreover, pre-configured images often force the user to run her application on specific cloud providers, inducing an undesirable vendor lock-in effect.

Arguably, the adoption of pre-configured images is still the most popular approach due to the lack of *integrated solutions* that support system designers and administrators throughout the entire process, ranging from the high-level declarative description of the application to the low-level deployment and configuration actions. In this paper we describe Blender, a software product maintained by Mandriva, which is based on the approach taken by the Aeolus project [7] that strives to overcome the limitations of using pre-configured images.

More precisely, Blender integrates three independent tools:

Zephyrus [10]. A tool that automatically generates, starting from a partial and abstract description of the target application, a fully detailed service oriented architecture indicating which components are needed to realise such application, how to distribute them on virtual machines, and how to bind them together. Zephyrus is also capable of producing *optimal* architectures, minimising the amount of needed virtual machines while still guaranteeing that each service has its needed share of computing resources (CPU power, memory, bandwidth, etc.) on the machine where it gets deployed.

Metis [25, 26]. A planner that generates a fully detailed sequence of deployment actions to be executed to bring an application to a desired configuration (e.g., as the one produced by Zephyrus). Plans are made of individual deployment actions like installing a software artefact, changing its state according to its internal life-cycle, provisioning virtual machines, etc. Metis relies on an

ad hoc planning algorithm that exploits service dependencies to prune the search space and produce the needed deployment steps very efficiently (i.e., provably in polynomial time). Metis could produce plans involving hundreds of components in less than one minute.

Armonic [27]. A collection of scripts and libraries that, starting from a knowledge base of information about available software artefacts, allows for the deployment of software applications and services on several Linux distributions. Each software artefact has a list of states, and each state performs actions to deploy and configure the associated software component on the target distribution.

By exploiting the above tools, **Blender** realises a framework that supports system architects and administrators all the way from the design phase down to the deployment on a cloud infrastructure. The present paper extends [10, 25–27] by offering a tighter integration among the three tools and by adding an actual user interface that turns **Blender** into a real, production-ready solution.

A declarative approach is adopted throughout **Blender**, according to which only a minimal amount of information needs to be initially given. For instance, it is sufficient to indicate the main services the application should expose to application users, plus non-functional requirements like the desired level of replication (for load balancing and/or fault tolerance) for critical service instances. From this initial information, **Blender** computes the complete architecture of the application and supports the administrator in the deployment phases, during which only context-dependent configuration variables needs to be manually instantiated.

Paper Structure. Section 2 presents **Blender** from the point of view of the users, by showing how to realise a real-life, moderately complex service-based cloud application: a replicated, load-balanced deployment of the WordPress blogging platform. Section 3 enters into more details, by showing what happens behind the scenes when **Blender** is used to realise the case study of Sect. 2. Section 4 points to the open source implementation of **Blender**. Before concluding, Sect. 5 reviews related literature and tools.

2 Deploying a WordPress Farm with Blender

We consider the deployment of a so-called “WordPress farm”, i.e., a load balanced, replicated blogging service based on WordPress.¹ A typical approach to deploy this kind of application is to rely on specific services listed in Table 1.

Instead, of adopting pre-configured virtual machines, on which instances of these software artifacts have been installed, our approach starts from reusable, abstract *descriptions* of these services, collected in the *Armonic* knowledge base. Only a limited amount of case-by-case information must be provided by the user. From these elements (the abstract description of the available software artifacts

¹ <https://wordpress.com/>.

Table 1. Software components used to deploy a WordPress farm

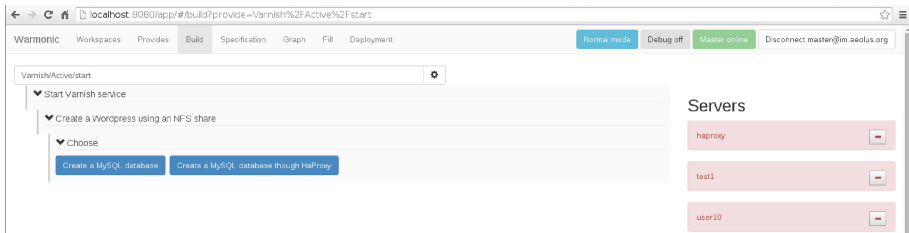
WordPress	a blogging tool based on PHP;
Galera Cluster	for MySQL: a multi-master cluster of MySQL databases synchronously replicated;
HAProxy	a load balancer for TCP and HTTP-based applications spreading requests across multiple servers;
Varnish	an HTTP accelerator designed for content-heavy dynamic web sites supporting dynamic load balancing;
HTTP Server	a software component serving web server requests;
NFS client/server	an application implementing a distributed file system.

and the additional specific user-defined information), **Blender** synthesises and then deploys the entire application.

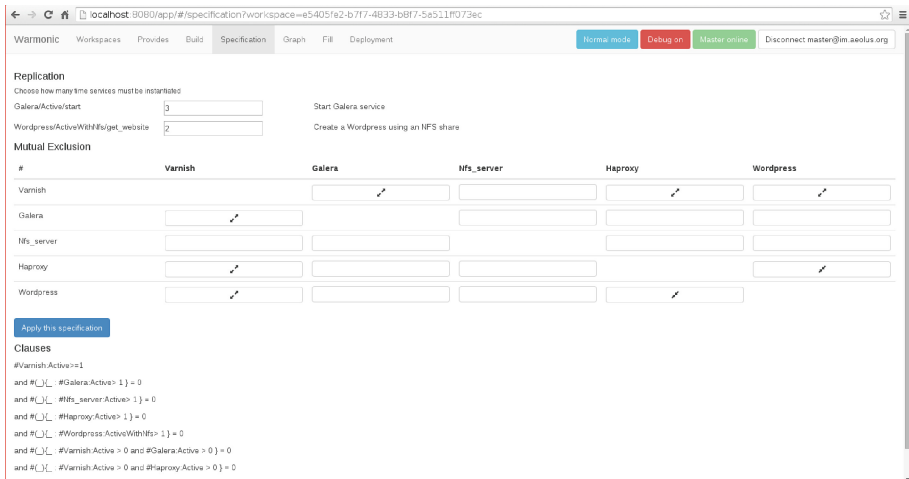
When executing **Blender**, the first piece of information the user will need to provide is an indication of the desired front-end service to be deployed, in this case the *Varnish* load balancer. Based on this initial piece of information, **Blender** will guide the user through an interactive question/answer phase, during which the main services needed to complete the application are chosen, and service-dependent additional information are asked to the user. The kind of information requested to the user in this second phase typically deals with desired installation policies, which usually vary on a case-by-case basis. For instance, as shown in Fig. 1a, once *Varnish* and *WordPress with NFS* is chosen, two different solutions for the database are proposed (i.e., single shared installation or multi-master replication based on *Galera*). As shown in Fig. 1b the user can then also specify that specific service pairs cannot be co-installed on the same virtual machine (e.g., *WordPress* cannot be installed with *Galera* for performance reasons) or that two services have to be co-installed (e.g., *WordPress* and *HAProxy* are installed on the same machine for fault tolerance reasons). This information cannot be automatically inferred, as it depends on specific properties like the expected workload, so user guidance is required.

Once these pieces of information are entered, **Blender** translates the description of the Armonic services into the Aeolus component-based model representations used by **Zephyrus** and **Metis**. In particular, **Zephyrus** synthesises the full architecture of the installation, indicates how many and which kind of virtual machines are needed, and distributes the services onto such machines. Subsequently, **Metis** computes the sequence of deployment actions needed to reach the final configuration produced by **Zephyrus**.

The computed plan is not ready to be executed yet, because some system-level configuration parameters are still missing (e.g., administrative passwords, credentials, etc.) and should be provided by the user. **Blender** asks the user for these information and, once all the configuration data is available, it proceeds to create the virtual machines computed by **Zephyrus** on the target OpenStack



(a) Selection choice: MySQL or a MySQLs replicated via HaProxy?



(b) Deciding and forbidding co-installation.

Fig. 1. User inputs for WordPress installation

infrastructure. Then, Blender uses Armonic to deploy and configure components by executing state changes, according to the Metis deployment plan.

In our example, during the interactive Q/A phase we have chosen Varnish to balance the traffic between 2 WordPress servers, NFS support, and 3 Galera instances. Moreover, we chose to inhibit co-installation of WordPress with Galera or the NFS Server, and to install HAProxy on every machine where WordPress is installed. The only additional piece of information asked by Blender as configuration data were the admin passwords for the DBs and HAProxy services.

The final architecture produced by Blender is depicted in Fig. 2. The installation requires 6 machines, 3 running Debian and 3 MBS (Mandriva Business Server). It took approximately 7 min to deploy such architecture on a simple OpenStack infrastructure deployed on an Intel Xeon server with 4 cores. The computation of the final configuration and the deployment plan was almost instantaneous: the execution of Zephyrus and Metis required less than a second while the most time consuming task was the deployment of the Galera

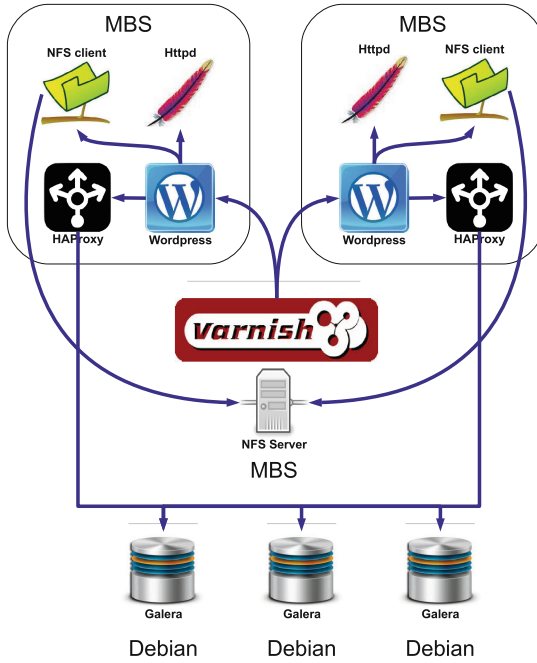


Fig. 2. Deployed WordPress farm architecture

Cluster that required 3 min and half (1 min and 10 s for every instance). The other services were deployed instead in less than a minute.

3 Blender Internals

As depicted in Fig. 3, Blender is intended to be used in combination with an XMPP server and an OpenStack cloud installation. Blender is realised as an XMPP client that wraps and combines the tools Zephyrus, Metis, and Armonic and exposes its functionalities via *ad hoc* commands.² Basically, such commands are used to launch Zephyrus, view the graph representing the computed final configuration, fill the configuration variables, and perform the deployment actions according to the plan produced by Metis. It is possible to interact with Blender via a Web user interface or the command line. An advantage of this architecture is that new elements can be added by wrapping them as simple XMPP clients. For instance, other IaaS offers can be easily added in addition to the currently supported OpenStack.

Blender relies on scripts that integrate Zephyrus, Metis, and Armonic following the execution flow depicted in Fig. 4. Such workflow requires two distinct

² <http://xmpp.org/extensions/xep-0050.html>.

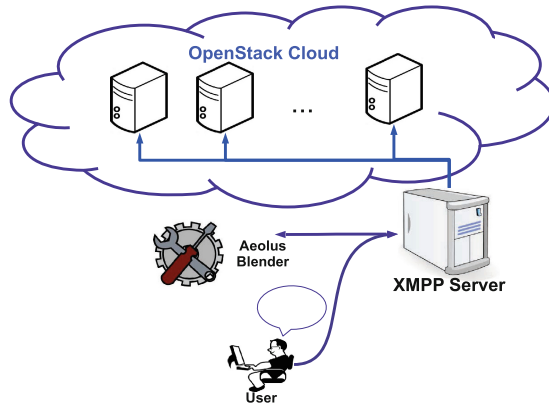


Fig. 3. Blender environment

inputs: an Armonic service repository, and a high-level description of the desired application to be deployed.

Armonic associates to every service a life-cycle that can be conceptually viewed as a state machine representing the different steps that need to be performed in order to deploy the service. For example, a service could have an associated state machine with 4 states: *not installed*, *installed*, *configured*, and *active*. Each state is usually associated to a collection of actions that need to be performed to enter into or exit each state, and actions that can be invoked on the service when a state has been entered. Technically speaking, states are implemented as Python classes, and actions are class methods. Each state has at least *enter* and *leave* hooks that are invoked when a state is entered and exited. Actions to be performed require the instantiation of a group of variables capturing information such as the required services, or the needed configuration values (with their default or optional values). In some cases, the required functionalities should be local when they must be provided in the same host where the component is deployed. For instance, in our running example, the NFS client is a local dependency of WordPress because an active WordPress needs an NFS client to be installed on the same machine.³

The first step of the Blender execution flow is querying the user to gather her desiderata. This task is performed by the *Builder* that asks the user for the services she wants to install, their desired replication constraints, and information about the need or impossibility to co-install onto the same host specific pairs of services.

When the user has entered all this information, the *Builder* queries the Armonic service repository and generates:

³ For more information related to Armonic services we refer the interested reader to [28].

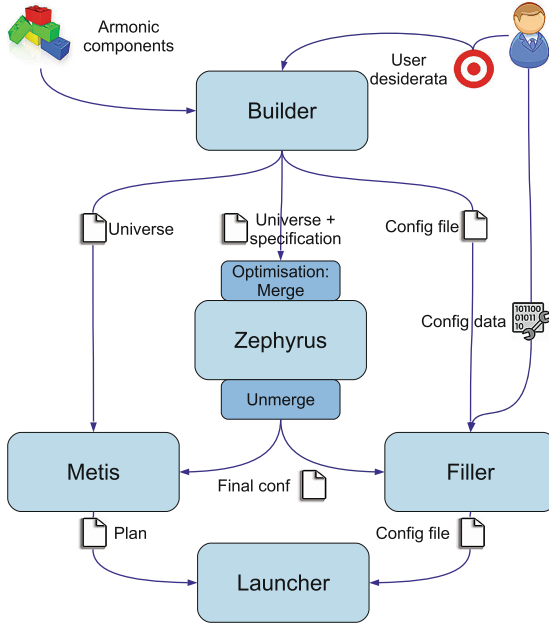


Fig. 4. Blender execution flow

specification file containing the encoding of the constraints that should be satisfied in the final configuration expressed in the specification language used by Zephyrus;

universe file containing the Aeolus component representations [11] of available services, in the JSON format used by both Zephyrus and Metis;

configuration data file containing indications about the system-level configured data needed to configure Armonic services. Some of them, if not already provided, will have to be entered by the user later on (e.g., credentials). Other data may be inferred from the configuration parameters of other components (e.g., WordPress can suggest a database name to its database dependency).

An excerpt of the specification file generated from user input for the running example is as follows:

```
Varnish:Active >= 1
and #(_){_ : #Galera:Active > 0 and
      #Wordpress:ActiveWithNfs > 0 } = 0
```

The first line requires a final configuration to have at least one Varnish service in the Active state. The second and third lines forbid the co-installation of Galera with WordPress. This is obtained requiring that the number of virtual machines having at least one Galera and one WordPress is 0.

The *universe* file is generated by encoding Armonic services into Aeolus components, which faithfully capture states and transitions. In Aeolus terminology,

methods exposed by states become provide ports. These methods and special state methods (e.g., *enter* and *leave*) can expose dependencies which become require ports. As an example, a graphical representation of the Aeolus model for the WordPress service of our example is given in Fig. 5. WordPress is depicted as a 5 state automaton, requiring the *add_database* functionality from the HAProxy to be configured, the *start* and *get_document_root* functionalities to be active, and the *mount* functionality from the NFS client to support the NFS. When active with NFS support, WordPress will provide the *get_website* functionality to other services.

Since the Aeolus model abstracts away from configuration data, these are stored in the *configuration data* file, which will be later used to perform deployment.

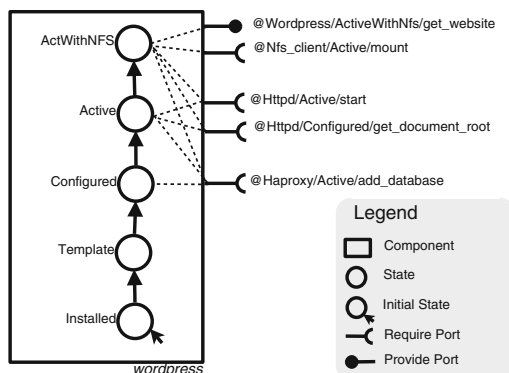


Fig. 5. Aeolus representation of the WordPress component

The universe file generated by the *Builder* is subsequently post-processed in order to merge together services that must be installed on the same machine. For instance, in our example, the WordPress services needs an NFS client to be installed on the same machine. These two services are therefore merged together obtaining a new service that consumes the sum of the resources. This simplifies the input of Zephyrus, reducing the number of services to be managed, thus speeding up the computation of the final optimal configuration, i.e., the one that uses the smallest number of virtual machines.

The solution computed by Zephyrus is then processed to decouple the services that were previously merged together. Indeed, while Zephyrus abstracts away from the internal life-cycles of the service, Metis needs to consider individual automata to compute the needed deployment actions. Metis then takes the post-processed output of Zephyrus and the original Universe file to compute a deployment plan to reach the final configuration.

At this point the user is asked to provide the missing configuration data for the final deployment. The configuration data file generated by the *Builder* is processed together with the output of Zephyrus to detect which services should

be installed and then fill the missing data querying the user if needed. This task is performed by a component dubbed *Filler* that uses several *Armonic* libraries to deduce configuration variables from default values when possible.

Once all the configuration information are filled in, the plan produced by Metis and the configuration data file are passed to the *Launcher*, a Python tool that acquires and bootstraps the virtual machines indicated in the output of Zephyrus using the OpenStack API, and transforms the abstract deployment actions generated by Metis into concrete actions that are sent to Armonic agents running on individual virtual machines.

4 Implementation

The complete toolchain presented in this paper is publicly available and released as free software, under the GPL license. **Blender** consists of approximately 5k lines of Python and is available from <https://github.com/aeolus-project/blender>. As **Blender** is an integrator, it has as software dependencies the tools it integrates:

- Zephyrus that amounts to about 10k lines of OCaml and is available from <https://github.com/aeolus-project/zephyrus>;
- Metis that amounts to about 3.5k lines of OCaml and is available from <https://github.com/aeolus-project/metis>;
- Armonic that amounts to about 5k lines of Python, plus glue code for service life-cycles written in shell script or Augeas and is available from <https://github.com/armonic/armonic>.

Screencasts showing the use of **Blender** to deploy different WordPress installations are available at <http://blog.aeolus-project.org/aeolus-blender/>.

5 Related Work

Currently, developing an application for the cloud is accomplished by relying on the Infrastructure as a Service (IaaS) or the Platform as a Service (PaaS) levels. The IaaS level provides a set of low-level resources forming a “bare” computing environment. Developers pack the whole software stack into virtual machines containing the application and its dependencies and run them on physical machines of the provider’s cloud. Exploiting the IaaS directly allows a great flexibility but requires also a great expertise and knowledge of both the cloud infrastructure and the application components involved in the process. At the PaaS level (e.g., [5, 18]) a full development environment is provided. Applications are directly written in a programming language supported by the framework offered by the provider, and then automatically deployed to the cloud. The high-level of automation comes however at the price of flexibility: the choice of the programming language to use is restricted to the ones supported by the PaaS provider, and the application code must conform to specific APIs.

To deploy distributed applications at the IaaS level, different languages with their deployment engines have been proposed. In this context, one prominent

work is represented by the TOSCA (Topology and Orchestration Specification for Cloud Applications) standard [32], promoted by the OASIS consortium [31] for open standards. TOSCA proposes an XML-like rich language to describe an application. Deployment plans are usually manually specified using the BPMN or BPEL notations, workflow languages defined in the context of business process modelling. Other similar deployment languages approaches are CloudML [17], the Fractal-based language extending the OVF standard [14], and approaches supporting the OASIS CAMP standard [30] such as Apache Brooklyn [3]. All these approaches allow a form of abstraction of the configuration to deploy. However, contrary to what can be done in **Blender**, the configuration to deploy have to be fully specified with all its configuration parameters and service dependencies. Moreover, due to their lack of a production-ready tool support, these approaches have seen a limited practical adoption so far. For this reason, as previously mentioned, the most common solution for the deployment of a cloud application is still to rely on pre-configured virtual machines (e.g., Bento Boxes [16], Cloud Blueprints [8], and AWS CloudFormation [2]).

Another common, but more knowledge-intensive solution, is to use configuration management tools which allows application managers to avoid some of the drawbacks of pre-configured images (e.g., lack of flexibility, lock-in mechanism) at the price of requiring a deep knowledge and expertise of the management tool and the configuration to realise.

One of the most similar approach to **Blender** is *Engage* [15], a tool that automatically generates the right order in which deployment actions should be performed to deploy some services. Engage avoids circular service dependencies and therefore the deployment plan can be generated by a simple topological sort of the graph representing the service dependencies. This is a significant limitation w.r.t. **Blender** because circular dependencies can arise in practice when, for instance, configuration information flow between services in both directions (consider, e.g., a master database that first requires the slave authentication and subsequently provides the slave with a dump of the database). Moreover, Engage does not provide a production-ready tool support.

Other commercial configuration management tools are instead Terraform [19], Juju [24], Cloudify [9], Rudder [29], and Scalr [37]. Terraform [19] is a configuration tool to describe both resources and services used to remotely execute a sequence of low-level deployment actions. However, it lacks a mechanism to describe the relationships between software services. Juju [24] is a tool and approach by Canonical, dedicated to the management of Ubuntu-based cloud environments. It is more a software orchestration framework than a proper configuration tool as it focuses on services and their relationships, to the detriment of many low-level aspects. Cloudify [9] is a software suite for the orchestration of the deployment and the life cycle of applications in the cloud. It is based on a meta language to describe a deployment plan and a monitoring software used to follow the application behaviour and to trigger a set of tasks to perform. Ruders [29] is an open source web solution dedicated to the production, automation and configuration of application deployment using CFEngine [6] and providing

real-time monitoring of the application trying to ensure its compliance using a rule base mechanism. Scalr [37] is an open source web application for cloud services management. Scalr uses the API of major cloud providers to deploy templates containing a Scalr agent that allows for fine grained interaction with supported services such as MySQL, Apache, etc.

All these commercial configuration management tools are used to declare the services to be installed on each machine and their configuration. However, contrary to **Blender**, the burden of deciding where services should be deployed, and how to interconnect them is left to the operator. Furthermore, no offering computes the final and optimal configuration starting from a partial specification, nor can devise the order in which deployment actions must be performed.

Other related works are ConfSolve [22] and Saloon [36]. ConfSolv is an academic approach that relies on a constraint solver to propose an optimal allocation of virtual machines to servers, and of application services to virtual machines. Saloon instead computes a final configuration by describing a cloud application using a feature model extended with feature cardinalities. Unfortunately, ConfSolve does not compute the actions needed to reach the computed configuration while Saloon automatically detects inconsistencies but, differently from **Blender**, it does not offer the ability to minimise the number of resources and virtual machines to be used.

Another relevant research direction leverages on traditional planning techniques and tools coming from artificial intelligence. In [4, 20, 21] off-the-shelf planning solvers are exploited to automatically generate (re-)configuration actions. To use these tools, however, all the deployment actions with their preconditions and effects need to be properly specified in a formalism similar to the Planning Domain Definition Language (the *de facto* standard language for planners). The **Blender** approach, on the other hand, relies on simpler and natural service descriptions (i.e., state machines describing the temporal order of the service configuration actions).

Finally, we would like to underline that **Blender** integrates various tools, some of which have been detailed elsewhere. Zephyrus has been presented in [10]. The present paper extends [10] in several ways: it integrates Metis to drive deployment on the basis of an actual deployment plan; it adds an actual user interface turning **Blender** into a real, production-ready solution; and it offers tighter integration among the three tools. Thanks to Metis, which supports the synthesis of infrastructure-independent plans, **Blender** could also be used with other deployment engines, while deployment as described in [10] relied on hard-coded internal mechanisms of Armonic. The new GUI supports the user in lively step-by-step visualisation of the effect of each deployment action. This functionality is effective if actions are executed in sequence. For this reason the current version of **Blender** serialises the actions synthesised by Metis (which, a priori, are parallelisable); future versions of **Blender** will consider parallel deployments by further improving its GUI.

Metis has been presented in [25]. The tool validation in that paper was done by using automatically generated descriptions of components. The integration

of Metis in Blender described in this paper, on the other hand, represents the validation of Metis on real use-cases.

6 Conclusions

We have presented Blender, a tool exploiting a configurator optimiser, an *ad hoc* planner, and a deployment engine to automate the installation and deployment of complex service-based cloud applications. Blender does not rely on predefined recipes, but on reusable service descriptions that are used as building blocks to synthesise a fully functional configuration satisfying the user desiderata. Blender is easy to use, comes with a web graphical interface, and requires as input just those specific configuration parameters that cannot be deduced from the service descriptions. It is an open source project started by Mandriva S.A., a software company specialised in Linux and open-source software providing innovative and easy to use offerings and business services to professionals. Mandriva with its research unit Innova is planning to exploit Blender offering a new server management solutions to speed up the current trend of migration of physical servers to cloud environments.

Since there is no standard benchmark for application deployment, as a future work we plan to define qualitative and quantitative evaluation mechanisms by first describing a series of deployment tasks that can later be used to evaluate both the improvements of future Blender versions and for comparison with possible future competitors. Moreover, as done in [12], we would like to compare the quality of the automatically generated deployment plans against those (manually) devised by DevOps. Since Blender always produces an optimal final configuration, its solution can be used to prove that an existing handmade solution is optimal. If instead the solutions differ due to the fact that some constraints were not specified or forgotten by the user, we may capture the missing requirements and then use them to ease and standardize the deployment of future similar deployment tasks.

Furthermore, we would like to reduce the deployment time of Blender by following the maximal parallelisable plan suggested by Metis. In this way, the deployment actions that are found to be independent may be executed in parallel. Moreover, noticing that replicated servers (e.g., the Debian machines containing the replicated database in our *WordPress* example) share part of their deployment plan, we would like to use live virtual machine cloning instead of re-creating instances that will end up being similar from scratch.

Further optimizing service deployment actions is outside the scopes of Blender. These actions are indeed intrinsic to the nature of the services to be deploy and depend just on their Armonic definition. However, we would like to tackle instead the time required by Zephyrus to compute the final optimal configuration. Indeed, as shown in [38] for some complex and large *WordPress* deployment scenarios, the computation of the optimal configuration may become the most computational intensive task of the toolchain. Even though for scenarios of reasonable size (for a typical professional *WordPress* installation) less than one

minute of computation is needed, in our biggest stress tests (i.e., in one case we required Zephyrus to compute a solution for a system having 103 software components distributed over 86 machines) we experienced computation times of more than 20 min. To reduce the time required by Zephyrus in these cases we therefore plan to adopt portfolio solvers (e.g., [1]) or exploit heuristics (e.g., local search techniques) to quickly get good but possibly sub-optimal solutions.

Finally we also plan to integrate other public IaaS solutions (such as Amazon EC2, RackSpace, or Google Compute Engine) directly as well as exploiting and interface with other services libraries and tools such as Juju [24] or Apache Brooklyn [3].

References

1. Amadini, R., Gabbrielli, M., Mauro, J.: A multicore tool for constraint solving. In: IJCAI, pp. 232–238 (2015)
2. Amazon. AWS CloudFormation. <http://aws.amazon.com/cloudformation/>
3. Apache Software Foundation. Apache Brooklyn. <https://brooklyn.incubator.apache.org/>
4. Arshad, N., Heimbigner, D., Wolf, A.L.: Deployment and dynamic reconfiguration planning for distributed software systems. *Softw. Qual. J.* **15**(3), 265–281 (2007)
5. Microsoft Azure. <http://azure.microsoft.com>
6. Burgess, M.: A site configuration engine. *Comput. Syst.* **8**(2), 309–337 (1995)
7. Catan, M., Di Cosmo, R., Eiche, A., Lascu, T.A., Lienhardt, M., Mauro, J., Treinen, R., Zacchiroli, S., Zavattaro, G., Zwolakowski, J.: Aeolus: mastering the complexity of cloud application deployment. In: Lau, K.-K., Lamersdorf, W., Pimentel, E. (eds.) ESOC 2013. LNCS, vol. 8135, pp. 1–3. Springer, Heidelberg (2013)
8. CenturyLink. Cloud Blueprints. <http://www.centurylinkcloud.com/products/management/blueprints>
9. Cloudify. <http://getcloudify.org/>
10. Di Cosmo, R., Lienhardt, M., Treinen, R., Zacchiroli, S., Zwolakowski, J., Eiche, A., Agahi, A.: Automated synthesis and deployment of cloud applications. In: ASE, pp. 211–222. ACM (2014)
11. Di Cosmo, R., Mauro, J., Zacchiroli, S., Zavattaro, G.: Aeolus: a component model for the cloud. *Inf. Comput.* **239**, 100–121 (2014)
12. de Gouw, S., Lienhardt, M., Mauro, J., Nobakht, B., Zavattaro, G.: On the integration of automatic deployment into the ABS modeling language? In: ESOC (2015)
13. DevOps. <http://devops.com/>
14. Etchevers, X., Coupaye, T., Boyer, F., De Palma, N.: Self-configuration of distributed applications in the cloud. In: CLOUD, pp. 668–675. IEEE (2011)
15. Fischer, J., Majumdar, R., Esmailsabzali, S.: Engage: a deployment management system. In: PLDI, pp. 263–274. ACM (2012)
16. Flexiant. Bento Boxes. <http://www.flexiant.com/2012/12/03/application-provisioning/>
17. Gonçalves, G.E., Endo, P.T., Santos, M.A., Sadok, D., Kelner, J., Melander, B., Mångs, J.-E.: CloudML: an integrated language for resource, service and request description for D-Clouds. In: CloudCom, pp. 399–406. IEEE (2011)

18. Google App Engine. <https://developers.google.com/appengine/>
19. HashiCorp. Terraform. <https://terraform.io/>
20. Herry, H., Anderson, P.: Planning with global constraints for computing infrastructure reconfiguration. In: CP4PS (2012)
21. Herry, H., Anderson, P., Wickler, G.: Automated planning for configuration changes. In: LISA. USENIX Association (2011)
22. Hewson, J.A., Anderson, P., Gordon, A.D.: A declarative approach to automated configuration. In: LISA, pp. 51–66 (2012)
23. IDC. Executive summary: a universe of opportunities and challenges (2012). <http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf>
24. Juju, devops distilled. <https://juju.ubuntu.com/>
25. Lascu, T.A., Mauro, J., Zavattaro, G.: A planning tool supporting the deployment of cloud applications. In: ICTAI, pp. 213–220. IEEE (2013)
26. Lascu, T.A., Mauro, J., Zavattaro, G.: Automatic component deployment in the presence of circular dependencies. In: Fiadeiro, J.L., Liu, Z., Xue, J. (eds.) FACS 2013. LNCS, vol. 8348, pp. 254–272. Springer, Heidelberg (2014)
27. Mandriva. Armonic. <http://armonic.readthedocs.org/en/latest/index.html>
28. Mandriva. Armonic, Lifecycle anatomy. <http://armonic.readthedocs.org/en/latest/lifecycle.html>
29. Normation. Rudder. <http://www.normation.com/en>
30. OASIS. Cloud Application Management for Platforms. <http://docs.oasis-open.org/camp/camp-spec/v1.1/camp-spec-v1.1.html>
31. OASIS. Organization for the Advancement of Structured Information Standards (OASIS). <https://www.oasis-open.org>
32. OASIS. Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0. <http://docs.oasis-open.org/tosca/TOSCA/v1.0/cs01/TOSCA-v1.0-cs01.html>
33. Opscode. Chef. <http://www.opscode.com/chef/>
34. PAC. Cloudindex study (2014). <http://www.cloudindex.fr/sites/default/files/PAC%20CloudIndex%20-%202014.pdf>
35. Puppetlabs. Puppet. <http://puppetlabs.com/>
36. Quinton, C., Pleuss, A., Le Berre, D., Duchien, L., Botterweck, G.: Consistency checking for the evolution of cardinality-based feature models. In: SPLC, pp. 122–131. ACM (2014)
37. Scalr Cloud Management. <http://www.scalr.com/>
38. Zwolakowski, J.: A formal approach to distributed application synthesis and deployment automation. Ph.D thesis, Univeristé Paris Diderot - Paris 7 (2015)

Analyzing Resource Behavior to Aid Task Assignment in Service Systems

Renuka Sindhgatta¹(✉), Aditya Ghose², and Gaargi Banerjee Dasgupta¹

¹ IBM Research-India, Bangalore, India
renuka.sr@in.ibm.com

² University of Wollongong, Wollongong, NSW, Australia
aditya.ghose@uow.edu.au, gdasgupt@in.ibm.com

Abstract. Service organizations increasingly depend on the operational efficiency of human resources for effective service delivery. Hence, designing work assignment policies that improve efficiency of resources is important. This paper explores the role that data (specifically service execution histories) can play in identifying optimal policies for allocating service tasks to service workers. Using data from the telecommunications domain, we investigate the impact of assigning similar and distinct tasks within the temporal frames of a day, across days and a week. We find that similar work, when done within a day, significantly improves the efficiency of workers. However, workers working on distinct tasks across days also have higher efficiency. We build a simulation model of the service system under study, to gain insights into the dispatch policy considering similarity and variety of tasks assigned. Our work demonstrates use of data to generate critical insights on resource behavior and efficiencies, that can further aid in improving task assignment to resources.

Keywords: Resource assignment · Task similarity · Task variety · Simulation model

1 Introduction

A *service system* as defined by Sphorer [9] is an important unit of analysis in support of understanding the operations of an organization. A service system comprises of resources (that include people, organizations, shared information, technology) and their interactions that are driven by a process to create a suitable outcome to the customer. Arguably, the most critical resources in a service system are human resources. Without loss of generality, we shall refer to these as Service Workers (SW). Unlike machines or equipment, the behavior and efficiency of service workers is highly variable, and contingent on factors such as the experience gained from repeated execution of similar tasks (and potentially negatively impacted by the well-known psychological pitfalls that accrue from a lack of variety). Understanding resource behavior is critical as the overall efficiency of the service system or the service organization largely depends on the resources.

Several large service organizations provide services that involve procedural and repetitive tasks i.e. tasks that do not require creativity and innovation. Service workers are commonly assigned or reassigned tasks based on a variety of criteria including experience, skills, their availability and sometimes other, often arbitrary, measures. It is important to note that in an environment that emphasizes the need for cost efficiencies, these traditional measures of suitability (of a service worker to perform a particular task) may not be sufficient and in some cases relevant. Instead, it is critical to identify work design and allocation policies that can improve the efficiency of human workers. In the context of work design and assignment, there are studies indicating distinct approaches: (a) assign similar work to a service worker where rhythmic and repeatable work will result in improved efficiency [13], (b) assign different tasks to provide variety of work to the service workers and improve their motivation and reduce boredom [10]. (c) balance both similar and variety of tasks assignment to the worker [10]. There are limited studies on evaluating the influence of work design and efficiency and using it to aid task assignment, especially in domain where workers are multi-skilled and not every one in a team can perform the task. For example, how does short term experience of a worker influence efficiency? Does variety of work have an impact or influence on day to day work efficiency? Does multi-skilling provide benefits of variety to a service worker? We aim to address some of these questions through this work.

In this paper, we study the data from a large service system, from the telecommunications domain, and evaluate the efficiency of service workers with respect to work done in a short time frame such as a single day, across days and weeks. That is, we study the impact of short-term experience on the efficiency of workers. Further, we describe a heuristic approach for an improvement in productivity of a worker, based on findings and make targeted assignment of workers to tasks. The approach considers similar and distinct work done by workers in immediate past to make future assignment. Evaluation of a heuristic assignment policy is performed using discrete event simulation that mimics the operations of the service system under study. This paper considers a common type of service system, where efficiency of service workers directly contributes to organizational productivity. Ultimately, our work serves to highlight the utility of performing such analysis to generate domain-specific, or organization-specific insights.

The outline of this paper is as follows: We discuss related work in Sect. 2. Next, we define key concepts, present our hypothesis and discuss our data collection i.e. the data used for our analysis, in Sect. 3. Section 4, presents our data analysis and model developed to support our hypothesis. In Sect. 5, we build a simulation model to evaluate the improvements possible when assignment is done considering the work done by resources prior to the task under consideration. We discuss validity of our results in Sect. 6. Section 7 concludes the paper.

2 Related Work

In this section, we outline the background of our study in the light of related work. We present the research trends in two specific areas related to our study.

2.1 Modeling Service Systems

A resource model as defined by Ramaswamy et al. [17], forms a key element in building a formal service delivery model. Resources having capabilities required by tasks of service delivery process, are assigned to the process, to enable its completion. Resources in service systems are humans, referred to as Service Workers (SW). Service system models define attributes of service workers, such as availability by considering shifts roasters and capability by defining a skill vector. A Work or Service Request (SR) arriving in the service system is defined by considering complexity, severity (or importance) and the minimum capability required to complete the work. Dispatching policies [2], with considerations to the tardiness, lateness and utilization of the resources have been evaluated with respect to various service system workloads, that assign a SR to one or more service workers. In their work, Diao et al. [6] present the first detailed model of a complex delivery system. A model for an optimal labor cost given complex constraints of resource availability, capability and service level is defined. In one of the recent studies on service systems [1], the authors discuss how teams can be formed in accordance with one of the following service delivery models: (a) Customer focused (b) Business Function focused and (c) Technology-focused. Here authors hint, that the choice of the delivery model organization should be based on multiple factors, one of which is the expertise or skill of knowledge workers. Further, study on organizing service systems with teams having multiple skills has been evaluated and compared to social compute units [5, 18]. In these studies, the operational efficiency of resources with specific capability, is considered to be homogeneous. Our work, extends from existing service system studies, and defines a dispatching model based on the insights gathered from the data, by defining allocation policies influencing resource efficiency.

2.2 Resource Behavior Analysis

Behavior of resources, when executing processes has attracted significant research interest in the recent years. In [22], common pitfalls associated with building simulation models, has been highlighted, that includes incorrect modeling of human resources. The authors emphasize incorrect representation or modeling of human resources as the cause of simulation models providing misleading outcome measures. Outcome measures refer to the average utilization of resources, average throughput or number of requests completed periodically, service quality that includes completing work within a specified target time. A process mining framework that can be used to detect outliers in resource behavior indicators (RBI) has been proposed in [16]. RBIs include metrics related to resource utilization, resource skills and productivity. The framework helps in time series analysis of indicators for each resource. In [19], the authors present an approach that uses historical data and illustrate variance in operational productivity of workers, for requests with different priorities and complexities. Variances in efficiency of workers are used to define policies for dispatching and optimally staff teams. Organizational behavior research to improve work design indicates two distinct

strategies of specialization and variety. In one of the recent studies by Staats et al. [20], the authors suggest that specialization or similar work during a single day improves the productivity of the worker while variety of tasks across days, helps in retaining the worker within the organization. The study has been carried out for a Japanese Bank where a large part of the process is automated and human resources involved in executing manual tasks of the process do not require any specialized skill or training. Narayanan et al. [10] analyze the degree to which task specialization enhances learning, and show that excessive exposure to task variety is an impediment to learning. Learning effects have been observed for repetitive tasks in manual, cognitive and knowledge-based work [12]. In [15], the authors present the idea that similar case instances in a row can be processed faster than randomly distributed case instances. Case instances that possess similar attributes are grouped together and distributed to resources at runtime. Depreciation or forgetting models and its effects has been studied by a few assignment models [11]. Learning and forgetting models help in identifying the assignment policies prior to workers achieving steady state of productivity. Hence, as indicated in much of the work done in the past, resource behavior has an important bearing on the efficiency and quality of a business process.

In this work, we seek to study how the experience gained in a shorter temporal frames, impacts worker efficiency. The study is conducted on a service system, in the telecommunications domain, requiring specialized skills. We further use the insights gained from the study to aid task assignment (or dispatch policy), and evaluate outcome measures of the service system.

3 Background

We now outline the context of our study by presenting the concepts of service system used in this study.

3.1 Service System

Work arrives into the system when customers request for a service. Work is defined by a work type and is further characterized by a set of capabilities required to complete the work. Resource(s) having the capabilities are assigned to the work for completing it. There are several dispatching policies that are used to assign work to a resource. In certain service systems, the task or work may require more than one resource, and is handed over to multiple resources to complete it. In this study we limit ourselves to scenarios where a single resource completes the work, as the service system that we study consists of tasks that requires a single SW to complete the work. We define key concepts underpinning the service system below:

Work Request or Task. Work requests constitute inputs to the service system and are handled by service workers. Typically, a work request (WR) is characterized by a work type. In this paper, we use task, request and work request interchangeably.

Skills. A finite set of skills pertaining to the domain defined by S .

WorkType. Work type categorizes a work request. There are a finite set of Work Types WT . There is one to many relation of work type to skills required for the work type defined by $w : WT \mapsto S$.

Service Workers. Service Workers are the human resources in the service system, who work on Work Requests. There a finite set of service workers SW . A one to many relation of service worker to skills possessed by the worker is defined by $s : SW \mapsto S$.

Service Time. Service time refers to the time a service worker spends to complete the work request. Hence, it is the time between the work request being assigned to the worker, to the time the service worker completes the request.

Work Arrivals. The arrival pattern of service requests is captured for finite set of time intervals T (e.g. hours of a week). That is, the arrival rate distribution is estimated for each of the time intervals in T , where the arrival rate is assumed to follow a stationary Poisson arrival process within these time intervals (one hour time periods) [2, 7].

Dispatching or Task Assignment. The task assignment is done by assigning a work request to a service worker with the necessary skills such that $w \cap s \neq \emptyset$. Hence, a SW with any one of the skills required for the work type of the work request can be assigned the work request.

An important consideration in assigning task to service workers is their availability and suitability. To evaluate the tasks that service workers work on and their operational efficiency, we introduce the following hypotheses that guide our investigations:

HYPOTHESIS 1: Doing similar work within a day has a significant influence on productivity of a service worker and its influence lasts for a day.

Consistent with the previous research [20], doing similar work helps worker perform certain steps in the process faster, that improves operational efficiency or reduces service time. We further, evaluate the influence of doing similar work in immediate past such as previous day and week to determine the temporal frame of influence.

HYPOTHESIS 2: Working on work requests of different work types, i.e. doing a variety of work has an influence the operational efficiency of a service worker.

Studies indicate that, variety of work has an influence on employee turnover. Task variety lowers levels of boredom [23] and increases job satisfaction [8, 20].

HYPOTHESIS 3: Multi-skilled service workers focus on a limited number of work types. The benefit of training on multiple skills diminishes with workers focusing on a smaller percentage of work types.

Multi-skilling has been recognized as a tool for increasing production flexibility [14]. Studies in the past indicate multi-skilling through cross-training in a manufacturing set up, to be beneficial but find greatest benefit when cross-training is minimal [4, 14].

3.2 Setting and Data Collection

The setting for our analysis is a large telecommunications service provider organization. The organization provides services for fixed line telephone, mobile telephone and broadband services. A process aware information system (PAIS) is used, where customers using service of the organization report problems related to the services e.g. internet speed, modem failures, phone lines not functioning etc. Depending on the problem, a work request is created with a specific work type. Technicians from the service provider organization are assigned to work on these work requests and resolve them. Each technician, goes to the customer site and resolves the issue. A large percentage of problems or requests are handled by a single technician ($\sim 94\%$). Once a technician completes the task, a new task is assigned. A technician spends time in traveling from one customer location to another location. We do not consider the travel time in our study. The service time is computed as the time spent between a technician reaching the customer's premises and time of completion of the request. Each technician has one or more domain skills that enables to address problems of specific type: problems related to cable management, plain telephone service, digital subscriber line etc.

The PAIS helps capture the time an issue was raised by the customer, the work type, the time at which the technician reached the customer site and the time when the technician solved the problem. A period of 3 months is analyzed with more than 89000 work requests served by 490 technicians. There are close to 30 work types that have less than 100 work requests. We drop these work requests from our study leaving us with 78,350 work requests served by 480 technicians. The distribution of natural logarithm of service time in minutes has a mean of 4.28 and standard deviation of 0.768 is shown in Fig. 1(a). Resources complete between 1 to 8 tasks in a day depicted in Fig. 1(b).

4 Data Analysis

This sections presents the models developed to support the hypothesis presented in Sect. 3.

4.1 Performance Improvement Doing Similar Work

For testing our hypothesis 1, we use linear regression models to understand the relationship and influence of work done by service workers within a day, previous day and week on the service times. The service time is the dependent variable in the model. We compute the following independent variables:

- Number of Prior Similar Tasks (SimilarTasks): The number of work requests having the same work type as the current task, completed before the task in the same day.
- Number of Prior Dissimilar Tasks (DissimilarTasks): The number of work requests having different work type as the current task, completed before the task in the same day.

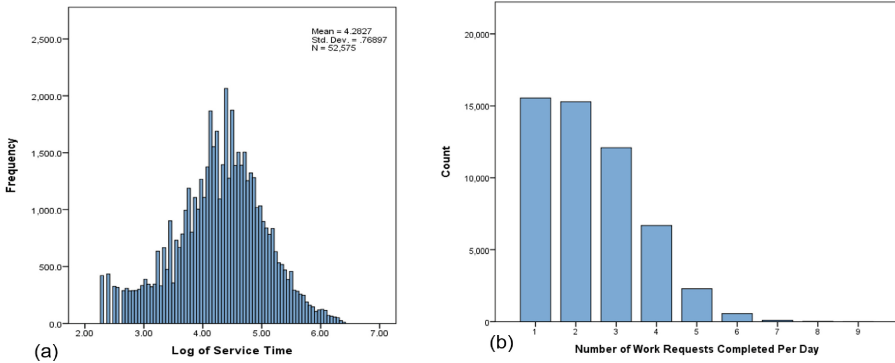


Fig. 1. (a) Distribution of log of service time (b) Histogram of work completed by resources per day

- Number of Similar Tasks Previous Day (PreviousDaySimilarTasks): The number of work requests having the same work type as the current task, completed on the previous day.
- Number of Similar Tasks in Week (WeekSimilarTasks): The number of work requests having the same work type, completed during the week.

These independent variables will help understand the temporal impact of working on similar tasks and the impact of working on different tasks in a single day.

We use multiple linear regression to understand the influence of independent variables on the dependent variable. We report the estimates of the coefficients of the independent variables along with their standard errors. The magnitude and the sign of the coefficients indicate the degree and directionality of influence of the corresponding independent variable on the dependent variable. The t value is the ratio of each coefficient to its standard error. Using this t value and the Student's t -distribution, the p value is calculated. If the p value is less than the significance level (usually taken to be 5% or 1%), the corresponding result is statistically significant. DF denotes the degrees of freedom. F is the Fisher F -statistic - is the test statistic for testing the statistical significance of the model. R^2 is the coefficient of determination - the ratio of the regression sum of squares to the total sum of squares, indicating the goodness of fit of the regression model. To compensate for over-fitting a model, adjusted R^2 is used that adjusts the R^2 value for the number of variables in the model. The objective of building the model, as indicated earlier, is to understand the influence of independent variables on the dependent variable.

We build a simple model, with the $\log(\text{service time})$ as the dependent variable and all the other independent variables indicating similar work and dissimilar work done by the technician prior to doing a particular task. We also control for the total work done by each technician every week as technicians or workers can have different volumes of tasks assigned. This is done by adding the total work done by each technician every week (TotalWorkInWeek) as an additional

variable into the regression model. The output of the model is shown in Table 1. This model shows that doing similar tasks in a day improves the service time. For example a worker doing a similar task twice in a day will have the service time of the second task reduced by a factor of 0.9 ($e^{-1*0.097} = 0.907$). Doing dissimilar tasks in the same day prior to the current tasks does improve the service time, but is lower than of doing similar tasks in the day. The influence of the tasks done in the previous day, on the service time is statistically insignificant as shown in the model. Hence, work done on the previous day, does not have any significant influence on the service time of the technician. Table 1 presents standardized estimate, that refers to how many standard deviations a dependent variable changes, per standard deviation increase in the independent variable. Standardized estimates help evaluate independent variables, that have a greater effect on the dependent variable. Doing similar tasks in the week improves service time, accounting for experience gained through the week. From the model, influence of doing similar tasks in a day and week has a large effect on the efficiency of the service worker. Therefore, this provides support for hypothesis 1.

Table 1. Multiple linear regression model showing service time based on similar tasks done in a day, dissimilar tasks done in a day, similar tasks done previous day and similar tasks done in a week.

	Estimate	Std. Estimate	Std. Err	t value	p-value
Intercept	4.282		0.013	319.6	<0.0001
SimilarTasks	-0.097	-0.073	0.005	-19.86	<0.0001
DissimilarTasks	-0.046	-0.040	0.004	-11.10	<0.0001
PreviousDaySimilarTasks	0.004	0.003	0.005	0.764	0.445
WeekSimilarTasks	-0.023	-0.098	0.001	-22.25	<0.0001
TotalWorkInWeek	-0.01	-0.051	0.001	-19.93	<0.0001
	DF = 78349	F = 1038.21	Adjusted R ² = 0.062		

4.2 Efficiency Improvement with Variety in Work

To study the impact of variety in the work done by service workers (hypothesis 2), we compute two independent variables:

- WorkerCapability is the number of work types a worker is capable of working on based on the skills possessed by the worker and skills required by the work type.
- WorkVarietyIndex is the ratio of the number of work types a service worker works on (on the job), and the WorkCapability. Valid values of WorkVarietyIndex would lie between [0,1]. A higher WorkVarietyIndex is indicative of a worker working on different work types, and hence, higher variety.

WorkVarietyIndex is incorporated into the existing model (of Table 1). We control for WorkerCapability because some workers may be trained on too few, or too many skills and hence, have very low or high WorkCapability respectively. The model with WorkVarietyIndex is shown in Table 2. Service workers

with higher WorkVarietyIndex have lower service time indicated by its negative coefficient. Hence, it supports our hypothesis 2 of work variety improving the operational efficiency of service worker.

Table 2. Multiple linear regression model showing service time based on similar tasks done in a day, dissimilar tasks done in a day, similar tasks done previous day and similar tasks done in a week and WorkVarietyIndex.

	Estimate	Std. Estimate	Std. Err	t value	p-value
Intercept	4.180		0.027	215.77	<0.0001
SimilarTasks	-0.098	-0.073	0.005	-18.196	<0.0001
DissimilarTasks	-0.044	-0.040	0.004	-13.895	<0.0001
PreviousDaySimilarTasks	0.002	0.001	0.003	0.419	0.675
WeekSimilarTasks	-0.005	-0.011	0.001	-3.847	<0.0001
TotalWorkInWeek	-0.001	-0.005	0.001	1.93	0.153
WorkVarietyIndex	-0.125	-0.051	0.014	-9.681	<0.0001
WorkCapability	0.001	0.092	0.000	36.976	<0.0001
	DF = 73940	F = 1266.386	Adjusted $R^2 = 0.107$		

4.3 Influence of Multi-skilling on Variety in Work

Multi-skilling allows workers to be more flexible addressing changes in demand, absenteeism and work assignment. We created a simple model to examine the relationship between variety in the work done by service workers, during the period of study (WorkTypes WorkedOn) and the variety in work a service worker is capable of working on by virtue of skills possessed (WorkType Capable). The result of the model is presented in Table 3. We use linear regression model without an intercept. The model shows that, workers are utilized on 21.7% of the work types they are capable of. This could be, due to workers possessing a large number of obsolete skills that may not have any demand. However, based on the model and results, hypothesis 3 of workers focusing on a limited number of work types is supported by Table 3.

4.4 Dispatching Considering Resource Behavior

Service time influences the number of requests completed per day or week by a service worker (throughput) and the time a resource is busy servicing the requests (utilization). Our model observations can be used to improve the dispatching rules or policies when assigning tasks to service workers. Algorithm 1 formally describes the policy. Initially, the minimum permissible queue length i.e. queueLenThreshold, is set to zero. queueLenThreshold is the number of requests that can be pending with the service worker. The dispatching policy checks for service workers with minimum permissible queue length and possessing the required skills. Among them, it finds a service worker, who has completed a minimum of 1 work request similar to the work type of the WR to be assigned. This is to account for assigning the work that is similar to previous completed

Table 3. Model predicting WorkTypes workedOn using WorkType Capable.

	Estimate	Std. Error	t value	p-value
WorkType Capable	0.217	0.00	911.7	<0.0001
	DF = 485	F = 882.647	Adjusted $R^2 = 0.646$	

work. If there are no service workers available, then the policy looks for service workers having a queue length of 1 and 2 by increasing the minimum permissible queue length, until the `MaxQThreshold` is reached. If it does not find any service worker, then the service worker with the least queue length is chosen. In the following sections, we refer to this policy as `SimilarWorkDispatch` policy. In the next section, we compare `SimilarWorkDispatch` policy to the policy of assigning tasks to a worker with suitable skills and the lowest queue length. We refer to the dispatching policy of assigning work to a SW with the required skill and lowest queue of pending requests as `MinimumQueueDispatch` policy.

Input: $WR, SWList$

Output: SW_{id}

$id = \phi;$

$queueLenThreshold = 0;$

$MaxQThreshold = 3;$

while $id = \emptyset$ OR $queueLenThreshold < MaxQThreshold$ **do**

$maxPreviousSimilarWR = 0;$

foreach $w_i \in SWList$ **do**

if $w_i.QueueLength == queueLenThreshold$ **then**

 // get number of completed requests matching WR workType ;

$wSimilarCount = getCompletedSimilarWorkInDay(WR);$

if $wSimilarCount > maxPreviousSimilarWR$ **then**

$id = w_i.id;$

$maxPreviousSimilarWR = wSimilarCount;$

end

end

end

if $id == \emptyset$ **then**

$queueLenThreshold = queueLenThreshold + 1;$

else

break ;

end

end

if $id == \emptyset$ **then**

$id = getWorkerWithLeastQueueLength;$

end

Algorithm 1. Work Similarity based Dispatching Policy considering previous similar work done by service workers

5 Simulation Based Experimentation

We describe the simulation set up that mimics the service system being evaluated. The simulation model enables us to compare overall performance of the service system using a dispatching policy that considers resource behavior, as described in Sect. 4.4 - SimilarWorkDispatch policy, and MinimalQueueDispatch policy. Each Work Request that comes into the system is assigned to a suitable service worker based on the dispatching policy. For the purpose of simulation, the following parameters are set.

- Work Arrivals: A finite set of time intervals for arriving work, denoted by T , containing one element for each hour of week. Hence, $|T| = 168$.
- Work Type: A finite set of WorkTypes are generated and each work request is associated with a work type when it is created.
- Worker Queue Length: Each service worker has a queue. The number of requests in the queue determines the load on the worker.
- Service Time: The service time of the worker used in the simulation model is based on the service system under study. For each work request, the number of prior similar work types completed by the service worker, within the day, is computed. The mean and the standard deviation of service time, for a specific number of prior similar tasks is computed. Figure 2 shows the plot of means of log service time varying with the number of similar tasks done within a day. In the simulation experiments, we compute the service time for each worker based on the (μ_s, σ_s) with s indicating the number of similar tasks done by the worker in the day.
- WorkVarietyIndex: The WorkVarietyIndex for simulation experiments uses a normal distribution with a mean, $\mu = 0.27$ and a standard deviation, $\sigma = 0.07$. Figure 2 shows the box plot of WorkVarietyIndex of the system under study. Each SW is assigned a WorkVarietyIndex based on the normal distribution with the mean and sigma values of the SS under study. The service time of a technician with higher WorkVarietyIndex is lowered by factor $(e^{-0.125 * WorkVarietyIndex})$. The coefficient -0.125 is taken from the regression model. WorkVarietyIndex is associated to each SW.

We build the service system model using AnyLogic simulation software [3,21] which supports discrete event simulation technique. We simulate up to 40 weeks of simulation runs. Measurements are taken at end of each week. No measurements are recorded during the warm up period of first four weeks. For our experiments, we consider request arrivals follow a Poisson model where the inter-arrival times follow an exponential distribution. In steady state the parameters that are measured include:

- Throughput or the number of work requests completed per week.
- Resource utilization: captures ratio of busy-time of a resource to the total time of the simulation run.

The simulation model is evaluated with the work arrivals derived from data. For the purpose of simulation, a smaller subset of the real-life data is used.

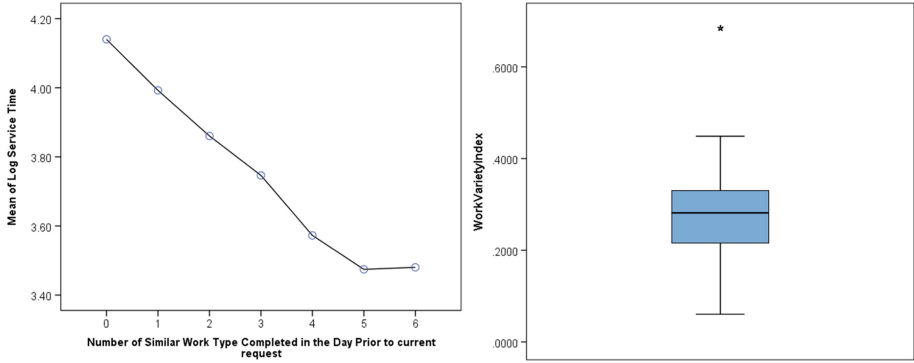


Fig. 2. Mean of Log Service Time with Similar Work done in the day, Box plot of WorkVarietyIndex

Each work is associated to one of the 15 work types similar to that of the service system under study. One hundred service workers are instantiated in the model. Each service worker is multi-skilled and is initialized with a specific WorkVarietyIndex. The results of the simulation experiments are presented in Fig. 3. Results compare two dispatching policies - MinimumQueueDispatch, SimilarWorkDispatch. The SimilarWorkDispatch policy is run for different value of MaxQThreshold values. As indicated in Fig. 3, SimilarWorkDispatch policy provides higher throughput when the MaxQThreshold = 2. At higher values of MaxQThreshold, the algorithm starts dispatching to workers with higher queue length. The throughput reduces as the requests wait in the queue of the service workers. Hence, for higher MaxQThreshold values, the gains from improved service time, by doing similar work, is offset by the wait time in the queue of the service worker. An additional simulation is run where workers do not have any variety in their work or the WorkVarietyIndex of service workers is set to 0 - SimilarWorkDispatchNoVariety. The observation is made to validate the improvement achieved by workers having variety in their tasks.

Figure 3 also compares the average resource utilization between these distinct dispatching policies. As the MaxQThreshold for SmartWorkDispatch policy increases, there are a few resources assigned the task (resources who have done similar work), while other resources remain free. Hence, the average resource utilization reduces. Hence, the SmartWorkDispatch policy is sensitive to the MaxQThreshold which should be set based on the evaluation of historical data for a service system.

Applying Insights from the study: The insights obtained from the simulation runs can be applied in practice to improve the efficiency of the service system. Important considerations that emerge from the experiment are: (i) Dispatching similar work provides substantial improvement in the performance of the service system, (ii) The gains due to the dispatching similar work policy should be evaluated based on the reduction in service time by doing similar work and the

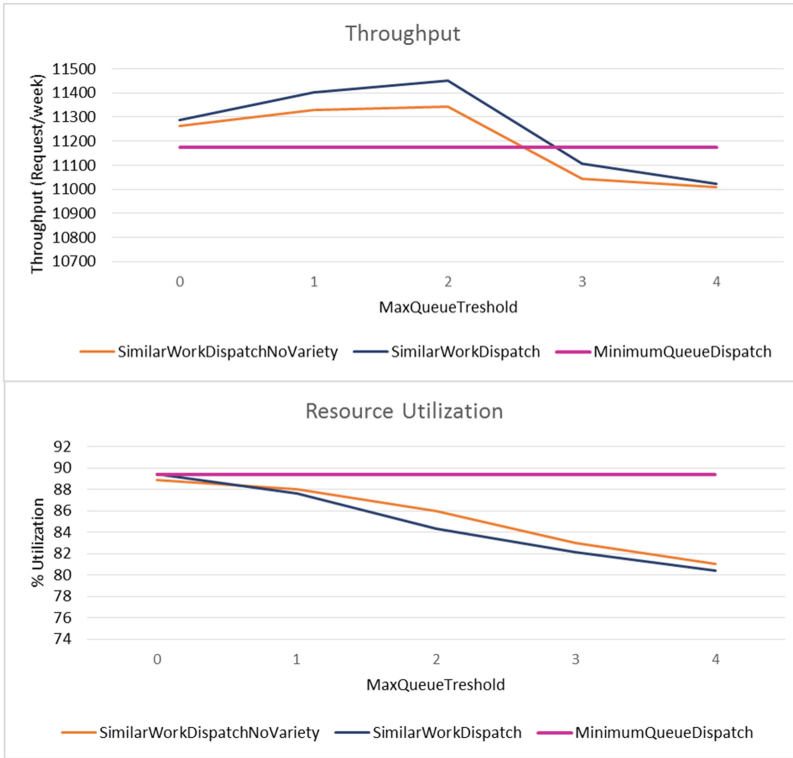


Fig. 3. Throughput and Resouce Utilization comparing (a) Similar Work based dispatch (b) Worker with minimum Queue length

wait time in the queue of busy service workers. (iii) As work variety improves the performance of the service system, variety in can be balanced by dispatching different work types across days and similar work within a single day.

6 Threats to Validity

There are some potential threats to validity for this work. It is extremely hard to predict the behavior of human resources working on tasks requiring specialized skills. The adjusted R^2 of the regression models were small, indicating that these effects highlight a small amount of the wide variance in the service time of the workers. However, our results are in line with some of the work done in the past [20, 23], in the broader context of organizational behavior and work design and do provide insights on the influence of similar and variety of work done on the service times of workers. In this system under study, data related to quality of work done was not captured. Hence service time has been viewed as an imperfect proxy measure of quality.

Further, we have studied a single, but large service system in this work. While, the work requests handled by the workers is repetitive, they are not done

in large volumes. The service levels for the requests are relaxed and the workers do not have any specific targets to finish in time. While insights can be drawn from our study, we do not claim that these results can be generalized in all instances. These results serve as the basis of using data driven approach for evaluating understanding resource behaviors in similar contexts and use them to improve task assignment.

7 Conclusion and Future Work

In this paper we studied the data from a large telecommunication service provider system. The impact of assigning similar and dissimilar tasks to a worker in a temporal frame of within a day, across days and a week was analyzed. From our results, we observe efficiency gains by a worker is significant when doing similar tasks in a day. Further, doing variety of work across days also improves efficiency. A simulation model was used to evaluate benefits of establishing a dispatching policy for task assignment. Through this work, we demonstrate, the value of such analysis in specific organizational contexts. In future, we would evaluate and analyze resource efficiency in other domains such are IT service management requiring specialized skills for completing tasks.

References

1. Agarwal, S., Sindhgatta, R., Dasgupta, G.B.: Does *One-Size-Fit-All* suffice for service delivery clients? In: Pautasso, C., Zhang, L., Fu, X., Basu, S. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 177–191. Springer, Heidelberg (2013)
2. Banerjee, D., Dasgupta, G.B., Desai, N.: Simulation-based evaluation of dispatching policies in service systems. In: WSC, pp. 779–791 (2011)
3. Borshchev, A.: The Big Book of Simulation Modeling. Multimethod Modeling with AnyLogic 6. Kluwer, Boston (2013)
4. Brusco, M.J., Johns, T.R.: Staffing a multiskilled workforce with varying levels of productivity: An analysis of cross-training policies*. *Decis. Sci.* **29**(2), 499–515 (1998)
5. Dasgupta, G.B., Sindhgatta, R., Agarwal, S.: Behavioral analysis of service delivery models. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 652–666. Springer, Heidelberg (2013)
6. Diao, Y., Heching, A.: Staffing optimization in complex service delivery systems. In: CNSM, pp. 1–9 (2011)
7. Diao, Y., Heching, A., Northcutt, D.M., Stark, G.: Modeling a complex global service delivery system. In: WSC, pp. 690–702 (2011)
8. Fried, Y., Ferris, G.R.: The validity of the job characteristics model: A review and meta-analysis. *Pers. Psychol.* **40**(2), 287–322 (1987)
9. Maglio, P.P., Vargo, S.L., Caswell, N., Spohrer, J.: The service system is the basic abstraction of service science. *Inf. Syst. E-Business Manage.* **7**(4), 395–406 (2009)
10. Narayanan, S., Balasubramanian, S., Swaminathan, J.M.: A matter of balance: Specialization, task variety, and individual learning in a software maintenance environment. *Manage. Sci.* **55**(11), 1861–1876 (2009)

11. Nembhard, D.A.: Heuristic approach for assigning workers to tasks based on individual learning rates. *Intl. J. Prod. Res.* **39**(9), 1955–1968 (2001)
12. Nembhard, D.A., Uzumeri, M.V.: Experiential learning and forgetting for manual and cognitive tasks. *Intl. J. Ind. Ergon.* **25**(4), 315–326 (2000)
13. Newell, A., Rosenbloom, P.S.: Mechanisms of skill acquisition and the law of practice. In: *The soar papers*, vol. 1, pp. 81–135. MIT Press, Cambridge (1993)
14. Park, P.S.: The examination of worker cross-training in a dual resource constrained job shop. *Eur. J. Oper. Res.* **52**(3), 291–299 (1991)
15. Pflug, J., Rinderle-Ma, S.: Dynamic instance queuing in process-aware information systems. In: *ACM 28th Symposium on Applied Computing*, pp. 1426–1433. ACM, USA (2013)
16. Pika, A., Wynn, M.T., Fidge, C.J., ter Hofstede, A.H.M., Leyer, M., van der Aalst, W.M.P.: An extensible framework for analysing resource behaviour using event logs. In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.) *CAiSE 2014*. LNCS, vol. 8484, pp. 564–579. Springer, Heidelberg (2014)
17. Ramaswamy, L., Banavar, G.: A formal model of service delivery. In: *SCC 2008*, vol. 2, pp. 517–520. IEEE, July 2008
18. Sengupta, B., Jain, A., Bhattacharya, K., Truong, H.L., Dustdar, S.: Collective problem solving using social compute units. *Int. J. Cooperative Inf. Syst.* **22**(4) (2013)
19. Sindhgatta, R., Dasgupta, G.B., Ghose, A.: Analysis of operational data for expertise aware staffing. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) *BPM 2014*. LNCS, vol. 8659, pp. 317–332. Springer, Heidelberg (2014)
20. Staats, B.R., Gino, F.: Specialization and variety in repetitive tasks: Evidence from a Japanese bank. *Manage. Sci.* **58**(6), 1141–1159 (2012)
21. XJ Technologies (2011). <http://www.xjtek.com/>
22. Van der Aalst, W.M.P., Nakatumba, J., Rozinat, A., Russell, N.: *Business process simulation: How to get it right*
23. Warr, P.B.: *Work, Happiness, and Unhappiness*. Routledge, New York (2007)

SenseX: Design and Deployment of a Pervasive Wellness Monitoring Platform for Workplaces

Rakshit Wadhwa, Amandeep Chugh, Abhishek Kumar, Mridula Singh,
Kuldeep Yadav^(✉), Sharanya Eswaran, and Tridib Mukherjee

Xerox Research Centre, Bangalore, India

{rakshit.wadhwa, amandeep.chugh, abhishek.kumar2, mridula.singh,
kuldeep.r, sharanya.eswaran, tridib.mukherjee}@xerox.com

Abstract. With the increasing number of desk jobs, workplaces have become the epicentre of several health risks. In this paper, we design and develop a pervasive wellness monitoring platform, *SenseX*, that uses a variety of devices and sensors to track physical activity level of employees in an organization. *SenseX* platform offers APIs which can be used by 3rd party applications to create services and applications, which can focus on specific interventions (e.g. to reduce prolonged sitting). We performed a real-world evaluation of the platform by deploying it in an IT organization for 6 weeks and observed longitudinal variations. We believe that *SenseX* platform helps in realizing the vision of “wellness as a service” in modern workplaces, enabling multitudes of different wellness services, which will be a key for sustained adoption of wellness programs.

1 Introduction

With the proliferation of computers and information technology in the last two decades, the number of desk jobs have grown at a phenomenal rate. In the United States (US), less than 20 % of private sector jobs have moderate levels of physical activity, decreasing by nearly 30 % compared to the early 60s [9]. Similarly, nearly 4 out of 5 people have desk jobs in the United Kingdom. A survey done in US reports shows that a typical worker spends 7.5 h sitting at work, 8 h sleeping, 4.5 h watching television or at home computer including leisure time, 1 h eating and only 3 h physically active or standing, i.e. sedentary for 21 h out of 24 h everyday [7]. Many studies have identified prolonged sitting as a high risk factor for severe health problems such as diabetes, cancer, heart attack, and stroke. For instance, [8] reports that adults who have moderate-to-high amounts of sitting time (four hours or more) have significantly higher cardio-metabolic risks compared to those who have lesser sitting time (less than three hours). Also, it is also found that the production of enzymes that burn fat declines by as much as 90 % after one hour of continuous sitting [5]. Additionally, it has been found out that excessive sitting results in depression, lower life expectancy, larger waist circumference and slower metabolism and over a term the harmful effects of sitting keep increasing. Surprisingly, researchers have found that regular exercise and balanced diet do not negate the adverse effects of prolonged sitting [4].

It is clearly evident that most of the sedentary behavior is found in workplace environments. Unarguably, they have become the epicenter of serious health risks and as a result, organizations have started investing in wellness programs. Workplace wellness is a \$6 billion industry in the United States alone where majority of organizations spending at least \$521 per employee per year [1]. Organizations are investing in these wellness programs to improve social, mental, and physical health of their employees as well as to reduce their healthcare payback costs. Most of these wellness programs incorporate costly wearable devices such as Fitbit, Nike Fuel Band, and Jawbone UP for activity tracking with physiological attributes. However, these wearables do not result in sustained adoption due to their obtrusiveness. For the users, it is more of an overhead to carry, wear and maintain an extra device. A study in US shows that more than 50% of consumers who owned an activity tracker stopped using them after 6 months [6]. Similar to activity trackers, standing workstations have also failed to have sustained impact and most users stop using those desks after one month of use [2]. Additionally, there are many playstore apps such as Motion24x7, Moves, Google Fit performs activity recognition using sensors available on the smartphone itself. However, these apps too fail to get a sustained adoption due to factors such as limited coverage i.e. users tend to leave their mobile phones stranded on desks while at work, and high battery consumption, which is due to nonstop sensing. Hence, there is a need for an unobtrusive, frugal, and pervasive platform for wellness monitoring and interventions in workplace environments.

In this paper, we design a *pervasive* wellness monitoring platform *SenseX* which leverages an employee's everyday devices and existing infrastructure (i.e. interconnected desktop/laptop, enterprise WiFi) for activity tracking and physiological measurements (i.e. heart rate). *SenseX* builds services to interface with different workstation devices (i.e. keyboard, mouse, webcam), workplace infrastructure (i.e. enterprise WiFi, calendar, BLE), and mobile phone sensors to sense activities and context of an employee. *SenseX* employs intelligent sensing approaches such as *triggered-sensing* i.e. offloading sensing load to infrastructure sensors whenever possible and *opportunistic-sensing* i.e. switching a sensor ON only when there is an opportunity to sense. Such approaches help in minimizing energy-consumption and increase sensing coverage. *SenseX* processes the device-specific measurements to infer fine-grained activities of a user as well as to extract high-order contextual information. A cloud-based *SenseX* service is used for the fusion of multiple device-specific activity profiles into a single profile, which is used for pushing appropriate interventions and notifications. Further, *SenseX* platform offers APIs which can be used by 3rd party applications to create services and applications, which can focus on specific interventions (e.g. to reduce prolonged sitting) to induce systematic behavior changes. We believe that *SenseX* platform helps in realizing the vision of "wellness as a service" in modern workplaces, enabling multitudes of different wellness services, which will be a key for sustained adoption of wellness programs. Specifically, this paper makes the following contributions:

1. First of its kind pervasive wellness monitoring platform which intelligently combine mobile, workstation, and infrastructure sensors to perform activity monitoring in workplaces.
2. Intelligent sensing approaches i.e. triggered-sensing to increase sensing coverage and reduce battery consumption. *SenseX* provided multitude of services and APIs to enable quick development of workplace-specific intervention and challenges.
3. A real-world deployment with 30 participants and detailed evaluation of *SenseX* platform with *StandUp* wellness challenge.

2 Related Work

The enhancement in the sensing capabilities of computerized devices along with the advancement in the area of ubiquitous and pervasive computing has resulted into the evolution of many fitness and healthcare applications/systems. We can broadly divide the previous work related to ours in three categories namely, Mobile-enabled sensing, Workstation-based sensing and Wellness platforms and services.

Mobile-Enabled Sensing. Modern-day smartphones are computationally powerful and at the same time are equipped with array of sensors, and thus have been widely used for day-to day sensing, activity recognition and health monitoring systems. Kwapisz et al. [10] proposed and developed a system for android devices, which could recognize, log and maintain simple physical activities like walking, jogging, sitting, standing etc. using phone based accelerometer. Similarly smartphones have been used widely for activity recognition [11, 13, 14]. Physiological parameter monitoring is yet another focus area in which researchers have exploited the capabilities of the smartphone. Aishwarya et al. [21] proposed smartphone based methodology to estimate the range of human blood pressure (BP) using Photoplethysmography. Similar to this [22] discuss a system which enables tracking the heart rate with user's finger tip placed on smartphone's camera. Sumida et al. [19] correlate the accelerometer, walking speed readings with the heartrate and estimate its variation while walking. There is no dearth of mobile-based systems designed with physiological parameter monitoring capability using integrated external sensors [25] and only internal sensors like camera etc. [23, 24].

Workstation-Based Sensing. Although workstations generally do not have any dedicated sensors, but many sensing hints can be obtained from inherent components like mouse, keyboard, touchpad, webcam etc. Thus researchers have exploited these sensing capabilities of workstation for measuring a number of health-related parameters like heart-rate, stress levels, emotional state etc. Sun et al. [15] suggest and demonstrate the use of mouse events to detect stress. Epp et al. [26] show that emotional states can be recognized using keystroke features. On the other hand [16–18] discuss and exhibit the ability of webcam equipped workstations to predict heart-rate.

Wellness Platforms and Services. The growing awareness and realization regarding the effectiveness of preventive healthcare has motivated the research community to design and develop several dedicated and ubiquitous healthcare platforms/systems. [28] discuss a service-oriented architecture platform for integration of health-data from various personal health devices. Li et al. designed and implemented a cloud-based platform for personal health sensor data management in a collaborative manner. Similarly [30] discuss the implementation of a scalable, robust cloud-based platform which can manage the semistructured, unstructured, and heterogeneous physiological signal effectively and can satisfy high concurrent requests from ubiquitous healthcare services.

SenseX builds upon the research work done in mobile and workstation-based sensing outlined above. However, there is lack of a platform that provides pervasive sensing capabilities such as *SenseX* and flexibility to create new interventions/challenges especially in workplaces.

3 System Design

Modern workplace environment consists of existing infrastructure of inter-connected desktops (i.e. laptop/PCs) equipped with web cameras, and employees carry their mobile phones with them. *SenseX* platform is designed to perform multi-modal sensing that encompasses existing workplace infrastructure as sensor hints along with mobile-based sensors to

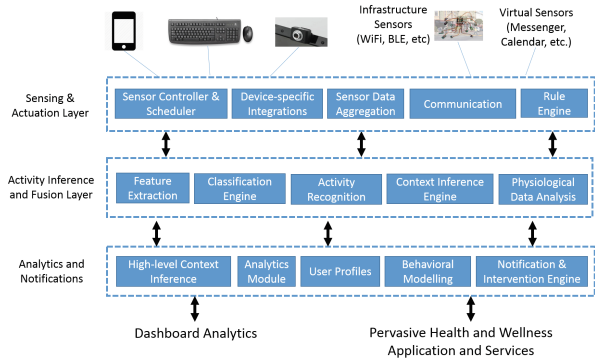


Fig. 1. System design of *SenseX* platform with different layers.

accurately track activity levels as well as wellness of an employee. *SenseX* platform follows a layered architecture involving three different layers designed for sensing raw data, and subsequently extracting high-level attributes to enable appropriate interventions. Figure 1 presents a snapshot of proposed three layered design with each layer consisting of a set of components. These components are part of the various services that run on end-user devices (i.e. mobile, workstation) and a cloud instance. *SenseX* follows a decentralized model where most of the sensing and processing of data is performed on end-user devices and aggregation is performed using a cloud instance. Further, the rationale of having layered design is that it enables flexibility of adding or removing layers as well as components by third party applications and services. These services can use activity tracking and context inference capabilities of *SenseX* and build their own wellness management and intervention solutions.

3.1 Sensing and Actuation Layer

This layer is responsible for interfacing with hardware devices and sensors to collect sensory measurements. Some of the primary components of this layer are device-specific integrations, sensor controller and scheduler, data collector, and a rule engine. *SenseX* using following device-specific integrations.

1. **Mobile Phone:** Mobile phone is equipped with a variety of different sensors such as location, light, proximity, accelerometer, gyroscope, microphone, etc. *SenseX* has a dedicated service running on the mobile phone, which can start/stop sampling any of these sensors based on the application requirements. All these sensors provide raw data values, which are later processed in higher layers to infer user-specific activity and contextual attributes. For example, accelerometer sensor data capture provides x, y, and z coordinate values corresponding to the motion in three axes/dimensions of a mobile phone while location sensor can provide the geo-coordinates or WiFi/GPS/Bluetooth fingerprint of user's current location.
2. **Workstation Devices:** Many people use desktop/laptop in their workplaces. *SenseX* runs a workstation service to capture different sensing cues from the workstation devices including activity on mouse/keyboard, name of foreground application (s), and webcam video feed. Based on the application requirements, *SenseX* service subscribe to one or more of these cues from the OS to collect raw sensor data. For example, an OS-based interrupt is generated whenever the keyboard or mouse is used. *SenseX* listens for such interrupts and records them with their respective time stamps. This data is processed in subsequent layers to infer high-order activities i.e. to find if a person was sitting or not in a given time interval. The main assumption is that if there is a steady stream of keyboard and mouse interrupts, the user is sitting continuously. Similarly, webcam-based video feed can be recorded as part of sensing, which could be used for simply detecting the presence of a person at workstation or to do more complex ones such as extracting physiological parameters.
3. **Infrastructure-Based Devices/Services:** Most of the organizations use different hardware sensors and software services as part of their day-to-day functioning. Some of these services include RFID card for access control, enterprise WiFi for ubiquitous network access, Bluetooth low energy (BLE) devices for tracking intra-organization movements, organization-wide calendar services for maintaining schedules, and messenger services for communication. *SenseX* provides customized services to gather sensor cues from available organization-wide sensors and services. For example, cues from calendar service can indicate the schedule of employees whereas enterprise WiFi or BLE can be used for localization of employees in the workplace environment.

Sensing Service Controller and Scheduler. Some of the above sensors are battery-constrained (i.e. mobile sensors) whereas some of them have near continuous power supply (i.e. workstation devices). One of the main responsibilities

of this layer is to control different sensors based on the application requirements, posed by higher layers. This module also identifies opportunities where battery-constrained sensors can be complemented with the data from infrastructure or workstation sensors. In a nutshell, it employs a novel *triggered-sensing* based approach to dynamically start/stop a device-specific service whenever there is an opportunity to use infrastructure-based sensors. The main idea behind triggered-sensing is to reduce sensing data redundancy as much as possible, thus improving battery life.

We present a use-case to demonstrate applicability and effectiveness of triggered-sensing based approach. For example, *SenseX* need to monitor a user Alice's sitting and standing patterns to provide an intervention for excessive sitting. Lets assume that Alice goes to her office by foot. While walking, *SenseX* mobile service will remain *ON* to keep track of Alice's activities. After, Alice reaches her workplace and starts using her workstation devices (i.e. keyboard, mouse), a trigger is generated by *SenseX* workstation service to mobile service for switching off mobile sensors as shown by the first trigger in Fig. 2. Meanwhile, workstation service keeps reporting the current activity (i.e. sitting) to the cloud instance. Whenever, there is an activity change detected by the *SenseX* workstation service i.e. there is no activity for a certain time, it starts sampling the webcam feed as represented by second trigger of Fig. 2. If it concludes that Alice is not present on workstation, then cloud instance send a trigger to mobile service to resume activity tracking.

Another advantage of triggered-sensing based approach comes in the form of increased sensing coverage where different sensing schemes can complement each other in terms of data collection. Consider the above scenario, Alice may place her phone on the table while working on the workstation and in such a case, a system using only mobile sensors will not be able to sense current state/activity. However, *SenseX* can sense the activity of Alice based on the usage of workstation devices. Further, the applicability of triggered-sensing based approach is not restricted to above described use-case only and it can be easily extended to a variety of other scenarios. For example, *SenseX* may use enterprise WiFi-based tracking for finding current location of a user rather than actively scanning mobile-based WiFi.

Rule engine component translates higher order activity inference requirements into low-level sensor mappings. For instance, if an application wants to track sitting-standing pattern of an employee, it enables mobile service and workstation-based sensors to track activity patterns. One of primary challenges faced by *SenseX* platform is in terms

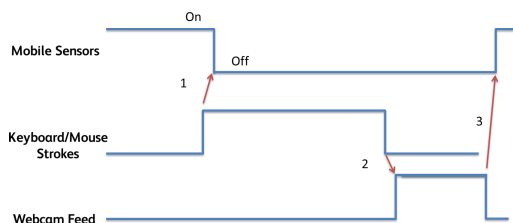


Fig. 2. An instance of triggered-sensing based approach employed in *SenseX*. Red color markers shows different triggers, which initiates other sensors (Color figure online)

of heterogeneity that includes different devices and lack of a standardized representation in sensing data. Sensing and actuation layer handles device heterogeneity by implementing different sensing interfaces for different devices and provide ability to control as well as collect sensory data from them. However, data heterogeneity still remains a challenge and higher layers as presented in Fig. 1 provide a solution for the same.

3.2 Activity Inference and Fusion Layer

The primary responsibility of this layer is to infer various high-level activities (i.e. sitting, standing, walking) as well as contextual attributes (i.e. inMeeting, Indoor/Outdoor, DeviceOnTable) from the raw data collected by sensing and actuation layer. After sensor/device specific inference, the different activities are fused together to create a wholistic activity profile for every user. Additionally, this layer hosts components to estimate physiological parameters from the sensing data.

Activity and Context Inference. This component aims to recognize high order activities from raw sensor measurements. Some of the activity inferences are straightforward i.e. if a person is typing on a keyboard/mouse or her presence is detected using webcam, her activity can be classified as ‘sitting’. However, some of the activity inferences need processing and in most of the cases require three different steps i.e. pre-processing of sensor data, extracting features, and applying a classification scheme to recognize the activity. Due to significant heterogeneity among sensed data values in *SenseX*, these processing steps need to be implemented distinctively for each device/sensor category. For example, accelerometer sensors provide X, Y, Z values representing motion in three axes/dimensions and it can be used to infer user activities such as sitting, standing, walking, climbing stairs, etc. The pre-processing steps involves dividing accelerometer data stream into segments (say x seconds) based on application and accuracy requirements. For each segment, it calculate set of statistical (mean, standard deviation, skewness, RMS), time-domain (integral, auto-correlation) and frequency-domain (spectral energy, spectral entropy) features. These feature values are then fed into a classification framework such as decision tree or SVM to recognize users’ activity.

Contextual information is an integral part of a wellness platform and it enriches the user-specific activity profile as well as notification delivery/intervention mechanism. For example, notifications could be smarter and will not be delivered when a person is in meeting. Some of the contextual information is easy to sense, for example, context information of a person being in a meeting or not, can be easily sensed from the calendar information. However, some of the contextual attributes such as detecting mobile phones’ presence in pocket require processing and fusion of multiple sensors data. For example, to detect whether the phone is in pocket or not, *SenseX* uses accelerometer sensor readings to find angle of inclination which is fused with proximity sensor information to classify whether

a phone is in pocket or lying flat while facing up or lying flat while facing down, etc. *SenseX* also interfaces with enterprise WiFi network and BLE technology to infer current location of the user [20].

Activity Fusion. Aforementioned components infer activity and context from device-specific sensor measurements and convert data representation from low-level to high-level activity or contextual constructs. *SenseX* utilizes different kinds of devices/sensors for activity and context tracking and most of higher order inferences (activity recognition) are performed on the device itself, while periodically reporting to *SenseX* cloud instance. This module perform fusion of activity constructs coming from two or more different devices/sensors. For example, the *SenseX* workstation service may detect that a person was sitting from 09:00 to 09:44 AM where as mobile service infers that the person was walking from 9:45 to 10:15 AM. This information is fused to make a single activity profile. In some of the cases, time information across two different activities may overlap due to errors in inferencing. In such cases, this module assigns higher priority to the infrastructure and workstation based sensors.

Physiological Analysis. Considerable research has been done to extract physiological parameters such as heart rate from activity data [19] as well as webcam-based video feed [18]. Further, there have been efforts to use webcam-based video feed to detect respiratory problems, emotions, etc. The focus of *SenseX* platform is to use existing research work as a pervasive mechanism to measure vital signs in day-to-day life. Heart rate measurement is an important vital parameter to diagnose various ailments such as anxiety, stress, cardio-vascular disease, etc.

Opportunistic Sensing. This component works in-conjunction with sensing service controller component to enable sensor tracking only when there is an opportunity detected. For example, physiological analysis does not work when there is motion w.r.t. subject's face in the video. In case of *SenseX*, there could be two kinds of opportunity-based sensing. Based on the detection of an opportunity, this component sends trigger to sensor controller module to enable tracking.

- **Context-directed/Detection:** In this case, detection of contextual attributes decide whether sensing should be enabled or not. For example, an application may require sensing of heart rate just after lunch time. To enable such sensing, required context is continuously tracked and a notification is issued to the user when there is an opportunity.
- **Sensing-directed/Prediction:** This is a completely automated and non-obtrusive approach where sensing is dependent on end-result i.e. automatically figuring out the instances where there is likely to be little motion so that heart rate tracking is possible using webcam. *SenseX* achieves this by continuously tracking system usage logs (i.e. mouse activity, keyboard activity, open applications) and using this information to predict the right moments with the help of a supervised classifier.

3.3 Analytics and Notifications

This layer performs high-level analytics on user-specific activity profiles and contextual attributes. High-level analytics is performed to detect *broad* patterns across an organization or a group of people as well as *personalized* behavior specific to a user. Based on these patterns, appropriate notifications can be generated and issued to users at opportune moments. This layer also contains a user profile component which tracks longitudinal data of users' response to the interventions/notifications, contextual information, privacy policies and the end result i.e. whether a user complied with the notification or not. The context tracking along with notification history captures specific behavioral attributes such as user *X* does not like to get disturbed while coding. This data in conjunction with behavior modelling component is used to employ the right persuasion strategies to motivate the user to comply with notifications. For example, an application dealing with prolonged sitting of employees can sense the sitting time using *SenseX* and accordingly, send notification in time to alert user to take break periodically. Based on the contextual attributes and longitudinal history, the notification may look like following:

“You have a meeting in “10” min and working from last 35 min. Please take a break now and walk for 2 min”.

4 Implementation Details

We developed *SenseX* platform with most of components described in Fig. 1. To demonstrate the efficacy of the platform for performing pervasive sensing and appropriate interventions, we developed an application *StandUp* to monitor activity levels of employees in an organization.

Mobile Service: It is a native OS service developed for Android devices due to its popularity and large market-share, especially in developing markets. *SenseX* mobile service is able to perform low-level sensory measurement i.e. accelerometer, proximity, microphone, etc. and contains activity inference algorithms to extract high-order activities and contextual attributes. *StandUp* is implemented as a native mobile application that accesses *SenseX* service to sense various activities (i.e. sitting, standing, walking, unknown, etc.). An activity sensed from mobile service is categorized as ‘unknown’ if the phone is kept on a flat surface (e.g. table). *StandUp* visualizes sensed activities in a day-based activity profile along with timeline as shown in Fig. 3a. Apart from day-based activity profile, users can see their weekly performance and an organization leaderboard where they can compare their performance with peers as shown in Fig. 3c. *StandUp* also uses *SenseX* intervention mechanism to provide engaging notifications to the users which provide “just-in-time” alerts for daily goals, compliance level etc.

Workstation Service: *SenseX* workstation service is implemented using Microsoft's .NET Framework 3.5 and supports all desktops/laptops running Windows OS. The service is configured to run whenever system is rebooted or



Fig. 3. a,b,c. Snapshots of StandUp Mobile Application d. Snapshot of workstation service ticker and notification

awakens after sleep mode. It supports functionality to track keyboard activity, mouse activity, webcam-based feed, and front ground application usage. *StandUp* uses the workstation service to track activity of an employee and builds a simple windows form based application to visualize such activity levels. *StandUp* application places a ticker on the users’ desktop to show the current station of user activity as shown in Fig. 3d. Similar to mobile service, *StandUp* application uses workstation service to provide timely notifications in case of prolonged sitting as shown in Fig. 3d.

Cloud-Based Instance: The server is hosted on a VM (virtual machine) provided by a public cloud service provider. The end-device services such as workstation service and mobile service communicate with the cloud-based instance using HTTP Get/Post requests. *SenseX* uses Google Cloud Messaging (GCM) to push updates on mobile devices. *StandUp* application uses APIs to provide a web-based dashboard which can be accessed by organization administrator to track installation, status of running services, application usage, organization leaderboard, and broad patterns.

5 Evaluation

SenseX is a platform designed for pervasive wellness monitoring platform in workplaces; consists of different services running across devices. As described in Sect. 4, *StandUp* application uses these services to provide activity monitoring and interventions for prolonged sitting in workplaces. The system evaluation of a platform such as *SenseX* is important to establish the effectiveness of different features (i.e. triggered-sensing) where as a typical wellness monitoring system will be questioned on how much can it contribute towards increasing activity levels, especially in a workplace setting. Specifically, for *SenseX* and *StandUp*, we conduct a mix of both *system-evaluation* (i.e. measuring impact of triggered-sensing, sensing coverage) and a *user-study evaluation* to capture the personal characteristic of users. We frame following research questions to guide both of these evaluation mechanisms.

- **R1:** How much sensing coverage is provided by different services of *SenseX*? How does sensing coverage vary across different users of the system?
- **R2:** What is the impact of triggered-sensing on the *SenseX* system? How much is the trigger-delay among different services? Does it help in saving energy for battery-constrained devices?
- **R3:** What are the activity patterns (i.e. continuous sitting time, daily steps) of employees working in a workplace environment? How do these patterns fair when put along organization wide leaders and slackers?
- **R4:** What is the impact of nudging an employee to take a stroll in case of prolonged sitting? How long does she take to respond to these notifications?
- **R5:** What is the impact of gamification and incentivizing the top performing employees? What is the observed difference between leaders and slackers in the system?

Deployment Details: We did a pilot deployment of proposed platform in an *IT* organization with a total of 30 participants (25 males and 5 females). The ages of participants ranged from 21 to 45 years, with a mean age of 29 years (male = 28, female = 31). All the participants used *StandUp* mobile and desktop application which were built upon *SenseX* services. The participants were given flexibility to specify the time intervals where activity tracking can be performed on desktop and mobile applications; default interval was set from 7AM to 7PM. To make the challenge engaging and competitive, we announced awards for top 3 weekly leaders chosen based on their performance over two metrics, i.e. average step count and average notification compliance score. Initially, we planned to run the deployment for 4 weeks but extended it for another 4 weeks keeping in mind engagement and interest of the participants. During the pilot, *SenseX* logged several parameters related to participants’ performance and interaction with the system. We analyzed the collected data to answer aforementioned research questions.

5.1 Sensing Coverage

The sensing coverage in the context of *SenseX* is defined as the fraction of time where activity monitoring is possible in a participant’s day-to-day life. *SenseX* uses mobile sensors along with contextual and environmental information, sensed by the workstation service. Using logs, we characterized activity monitoring w.r.t. sources i.e. mobile, workstation, or unknown. An activity is said to be ‘unknown’ if mobile device happened to be placed on a flat surface and there was no activity sensed by the workstation service. Across all the participants in our deployment study, we observed that mobile service sensed nearly 50 % of the total time, nearly 25 % of the time was monitored by the workstation service and rest of the time was categorized as unknown as can be inferred from Fig. 4c. This brings an interesting observation that mobile-based wellness applications such as Moves and Google Fit are not able to track activities for more than 50 % of time. Further, Fig. 4c shows the sensing coverage pattern for the subset of participants who had both mobile and desktop services actively running throughout

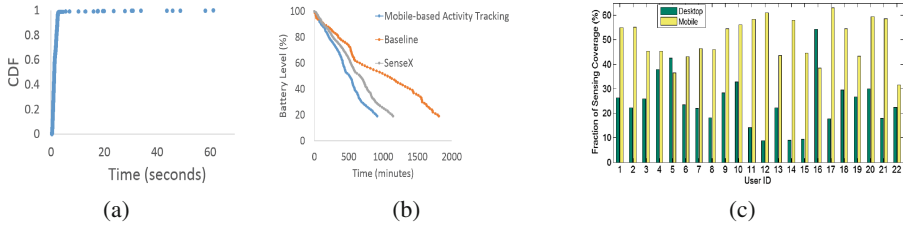


Fig. 4. a. Measured trigger-delay between mobile service to workstation service b. Battery decay comparison for both mobile-only activity tracking approach and *SenseX* based triggered-sensing with baseline c. Comparison of sensing coverage for mobile and workstation based services across different participants

the deployment study. We observe that for some users (ID: 5 & 16), more information is obtained from workstation service as compared to mobile sensors. It could be due to limited monitoring because some of the users enabled activity tracking during workplace timings only. Our analysis show that workstation-based service increases the sensing coverage to a large extent especially during workplace timings.

5.2 Triggered-Sensing and Battery Consumption

Section 3 presents triggered-sensing approach to offload sensing responsibilities to workstation service when there is an opportunity. In such cases, workstation service with the help of cloud instance need to send a trigger to mobile service indicating that it should stop sensing. We use push-based notification system to send trigger to a mobile service instead of a pull-model where it has to continuously look for a trigger by sending repeated request to the cloud instance. It is important to characterize the time to send this trigger from workstation service till it reaches to mobile service, represented as *trigger delay*. We emulated trigger-delay using *SenseX* platform where a total of 1200 requests were sent from workstation service to mobile service for a duration of 8 h. We present the observed trigger-delay in Fig. 4a, most of the time the delay was less than 10 s. Similarly, we compared energy consumption of *SenseX* activity tracking services w.r.t. only mobile-based tracker and baseline as shown in Fig. 4b. In mobile-based tracking approach, a continuous activity tracker that uses accelerometer ran on a MotoE phone till the battery reached to a low level where as baseline shows the battery decay without any external service. We found that *SenseX* using its triggered-sensing approach could increase the battery life time by nearly 25% of the time. From our experimental results, we conclude that triggered-sensing works in near real-time and helps in increasing sensing coverage as well as result in significant energy-savings.

5.3 Workplace Activity Patterns

StandUp challenge was designed to increase the overall activity level of employees at a workplace. Figure 5a presents the CDF of the number of steps for each

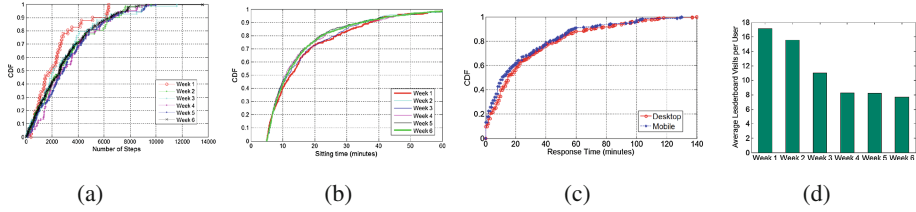


Fig. 5. a. CDF of day-wise steps across different participants b. CDF of sitting time sessions across different weeks c. CDF of the response time observed for all the notifications during the deployment d. Average number of leaderboard hits per user each week during the deployment

week across all participants. We see a clear shift among the first and sixth week, indicating that more people have increased their number of steps over the course of the pilot. Figure 5b shows the *CDF* of the sitting time of users, i.e., the time for which a user sits continuously at a stretch. We see that the duration of sitting segment of the users have marginally decreased from week 1 to week 6, which indicates that the users started adhering to the notifications and had been avoiding sitting continuously for longer durations. We conclude that *StandUp* challenge was effective in increasing the overall activity level among employees. Considering that, there was no effective way to capture baseline (i.e. participants’ performance before intervention), we have only provided performance comparison on week-to-week basis.

5.4 Effect of Notifications

One of the important aspects of *StandUp* challenge was to continuously monitor the sitting-standing pattern of the people and send notifications if they sit for more than a threshold time t . In case of *StandUp*, t was set to 40 min based on the prior medical studies and repetitive notifications were pushed every 5 min. The participants also had an option to stop the notification temporarily if they were busy. We compute a compliance score which is an indicator of percentage of time a participant has acted on notification or remain active at least 2 min in an hour. Figure 5c shows the *CDF* of response time for both workstation and mobile notifications. Response time is the time between when a notification is pushed and the user moves out of his sitting position (and proceeds to stand or walk). We see that for nearly half of the notifications, participants have acted in less than 10 min. We also observe that there is a slight preference of mobile-based notifications over workstation ones. Also, long tail of larger response times are observed because many time participants did not carry their mobile phone when they took a break.

5.5 Effect of Incentives and Gamification

With the help of *SenseX* analytics capabilities, *StandUp* maintained an organization-wide leaderboard which could be accessed by the participants. The

leaderboard was established based on two metrics i.e. average step count and compliance score. At the end of each week, cash prizes worth 15 USD each were announced for the top 3 leaders. We did not collect any qualitative data to measure the impact of leaderboard or incentives. However, we collected quantitative data on how many times leaderboard page was accessed by different participants during the deployment and we used that as a proxy to capture the interest of participants. Figure 5d presents the average number of leaderboard hits per user each week, (i.e., the number of times the leaderboard was visited by a user). We see that initially, it was visited two times per day on average, and the frequency gradually reduced as the initial excitement of the pilot settled in. However, the leaderboard visits have remained stable over the last 3 weeks, converging at around one visit every day by each user on an average.

Next, we analyzed the performance attributes of leaders and slackers during the deployment study. The participants above the 95th percentile of step count are taken as the leaders and those below the 5th percentile of step count are taken as the slackers. It may be noted that the set of leaders and slackers may vary in each week. We see a clear correlation between the leaders/slackers and the average sitting time, response time, and leaderboard hits, i.e., the step count-based leaders also feature in the above-average category for leader board hits, and below average (i.e., shorter duration) for sitting time and response time, and vice versa for the slackers.

6 Discussion

Modern day workplace forces a sedentary lifestyle that involves reduced physical activity and prolonged sitting, which poses a risk of severe ailments including diabetes, cancer, heart attack and stroke. Current wellness solutions are costly, dependent on wearable devices, which do not have a sustainable impact. We designed and developed a comprehensive wellness monitoring platform i.e. *SenseX*, which intelligently uses combination of mobile, workstation, and infrastructure sensors to do pervasive activity and physiological monitoring in workplaces. With multitude of different services, *SenseX* enables easy creation of different wellness interventions and challenges that can drive sustained behavioral changes among employees. As an example, we created a wellness challenge i.e. *StandUp* to increase the activity levels of employees in an organization. We ran the challenge for nearly 6 weeks and observed several key insights i.e. increased daily step count across weeks, reduced sitting sessions, positive effect of gamification/incentives, etc. From a systems' evaluation perspective, *SenseX* uses *triggered-sensing* approach to increase the sensing coverage by nearly 25% and reduced the battery consumption considerably.

Through its pervasive sensing capabilities, we believe that *SenseX* helps in realizing the vision of “wellness as a service in modern workplaces, enabling plethora of different wellness services and interventions”. We provide a sample of some of the other challenges/interventions that can be created using *SenseX*.

1. Vitamin *D* Challenge: IT workers suffer from lack of vitamin *D*, which is obtained by sun-exposure. A service can be created using *SenseX* activity monitoring and contextual inference capabilities that infer the presence of a person in outdoor and uses weather data to estimate the vitamin *D* exposure.
2. Healthy Meetings: A significant amount of sitting time in an IT workplace happens during the meetings. *SenseX* can be used to create a challenge to have meetings where people opt to stand, which can be compared across different departments in a workplace.

In essence, *SenseX* can help in creating many such challenges, which will drive sustained adoption of wellness programs. The layered architecture of *SenseX* can be used to enhance specific components, for example, accurate behavior modelling will help in enhancing the response time of the platform notifications. Similarly, pervasive sensing capabilities of *SenseX* could be enhanced to measure conditions such as sensing stress using keyboard and mouse activity.

References

1. Employee Wellness Program. <https://hbr.org/2010/12/whats-the-hard-return-on-employee-wellness-programs>
2. <http://ergo.human.cornell.edu/CUESitStand.html>
3. <http://usatoday30.usatoday.com/news/health/medical/health/medical/cancer/story/2011-11-03/Prolonged-sitting-linked-to-breast-cancer-colon-cancer/51051928/1>
4. <http://www.runnersworld.com/health/sitting-is-the-new-smoking-even-for-runners>
5. <http://blogs.hbr.org/2013/01/sitting-is-the-smoking-of-our-generation/>
6. <http://endeavourpartners.net/assets/Wearables-and-the-Science-of-Human-Behavior-Change-EP4.pdf>
7. Ergotron JustStand Survey and Index Report - JustStand.org. <http://www.juststand.org/portals/3/literature/SurveyIndexReport.pdf>
8. Staiano, A.E., Harrington, D.M., Barreira, T.V., Katzmarzyk, P.T.: Sitting time and cardiometabolic risk in US adults: associations by sex, race, socioeconomic status and activity level, *British J. Sports Med.* <http://www.ncbi.nlm.nih.gov/pubmed/23981954>
9. Church, T.S., Thomas, D.M., Tudor-Locke, C., Katzmarzyk, P.T., Earnest, C.P., Rodarte, R.Q., Martin, C.K., Blair, S.N., Bouchard, C.: Trends over 5 decades in US occupation-related physical activity and their associations with obesity. *PloS One* **6**(5), e19657 (2011)
10. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Activity recognition using cell phone accelerometers. *ACM SigKDD Explor. Newsl.* **12**(2), 74–82 (2011)
11. Brezmes, T., Gorricho, J.-L., Cotrina, J.: Activity recognition from accelerometer data on a mobile phone. In: Omatu, S., Rocha, M.P., Bravo, J., Fernández, F., Corchado, E., Bustillo, A., Corchado, J.M. (eds.) *IWANN 2009, Part II. LNCS*, vol. 5518, pp. 796–799. Springer, Heidelberg (2009)
12. Hache, G., Lemaire, E., Baddour, N.: Mobility change-of-state detection using a smartphone-based approach. In: 2010 IEEE International Workshop on Medical Measurements and Applications Proceedings (MeMeA), pp. 43–46. IEEE (2010)

13. Khan, A.M., Lee, Y.-K., Lee, S., Kim, T.-S.: Human activity recognition via an accelerometer-enabled-smartphone using kernel discriminant analysis. In: 2010 5th International Conference on Future Information Technology (FutureTech), pp. 1–6. IEEE (2010)
14. Zhang, S., McCullagh, P., Nugent, C., Zheng, H.: Activity monitoring using a smart phone's accelerometer with hierarchical classification. In: 2010 Sixth International Conference on Intelligent Environments (IE), pp. 158–163. IEEE (2010)
15. Sun, D., Paredes, P., Canny, J.: MouStress: detecting stress from mouse motion. In: Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems, pp. 61–70. ACM (2014)
16. Poh, M.-Z., McDuff, D.J., Picard, R.W.: Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. *Opt. Express* **18**(10), 10762–10774 (2010)
17. Mestha, L.K., Kyal, S., Xu, B., Lewis, L.E., Kumar, V.: Towards continuous monitoring of pulse rate in neonatal intensive care unit with a webcam. In: Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE, pp. 3817–3820. IEEE (2014)
18. Wei, L., Tian, Y., Wang, Y., Ebrahimi, T., Huang, T.: Automatic webcam-based human heart rate measurements using laplacian eigenmap. In: Lee, K.M., Matsushita, Y., Rehg, J.M., Hu, Z. (eds.) ACCV 2012, Part II. LNCS, vol. 7725, pp. 281–292. Springer, Heidelberg (2013)
19. Sumida, M., Mizumoto, T., Yasumoto, K.: Estimating heart rate variation during walking with smartphone. In: Proceedings of the 2013 ACM International Joint Conference on Pervasive and ubiquitous computing. ACM (2013)
20. Balaji, B., Xu, J., Nwokafor, A., Gupta, R., Agarwal, Y.: Sentinel: occupancy based HVAC actuation using existing WiFi infrastructure within commercial buildings. In: Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, p. 17. ACM, November 2013
21. Visvanathan, A., Banerjee, R., Dutta Choudhury, A., Sinha, A., Kundu, S.: Smart Phone Based Blood Pressure Indicator. In: MobileHealth, 11–14 August 2014, Philadelphia, PA, USA (2014)
22. Pal, A., Sinha, A., Choudhary, A., Chattopadhyay, T., Visvanathan, A.: A robust heart rate detection using smart-phone video. In: MobileHealth 2013, Proceedings of the 3rd ACM MobiHoc workshop on Pervasive wireless healthcare, pp. 43–48 (2013)
23. Grimaldi, D., Kurylyak, Y., Lamonaca, F., Nastro, A.: Photoplethysmography detection by smartphones videocamera. In: Proceedings of IEEE International Conference IDAACS 2011, pp. 488–491, September 2011
24. Scully, C.G., Lee, J., Meyer, J., Gorbach, A.M., Granquist-Fraser, D., Mendelson, Y., Chon, K.H.: Physiological parameter monitoring from optical recordings with a mobile phone. *IEEE Trans. Biomed. Eng.* **59**(2), 303–306 (2012)
25. Kang, S., Kwon, S., Yoo, C., Seo, S., Park, K., Song, J., Lee, Y.: Sinabro: opportunistic and unobtrusive mobile electrocardiogram monitoring system. In: Proceedings of HotMobile, Santa Barbara, CA (2014)
26. Epp, C., Lippold, M., Mandryk, R.L.: Identifying emotional states using keystroke dynamics. In: CHI 2011, Vancouver, BC, Canada, 7–12 May 2011
27. Zeng, L., Hsueh, P., Chang, H.: Greenolive: an open platform for wellness management ecosystem. In: IEEE/INFORMS International Conference on Service Operations and Logistics (SOLI 2010) (2010)

28. Leel, S.-H., Song, J.H., Ye, J.-H., Lee, H.J., Yi, B.-K., Kim, I.K.: SOA-based integrated pervasive personal health management system using PHDs. In: Proceedings of the 4th International Conference on Pervasive Computing Technology Healthcare, Munich, Germany, p. 14, March 2010
29. Li, Y., Guo, L., Wu, C., Lee, C.-H., Guo, Y.: Building a cloud-based platform for personal health sensor data management. In: IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI), pp. 223–226 (2014)
30. He, C., Fan, X., Li, Y.: Toward ubiquitous healthcare services with a novel efficient cloud platform. *IEEE Trans. Biomed. Eng.* **60**(1), 230–234 (2012)

Opportunities for Process Improvement: A Cross-Clientele Analysis of Event Data Using Process Mining

R.P. Jagadeesh Chandra Bose^(✉), Avantika Gupta, Deepthi Chander,
Ajith Ramanath, and Koustuv Dasgupta

Xerox Research Center India, Bangalore 560103, India
{jagadeesh.prabhakara,avantika.gupta,deepthi.chander,
ajith.ramanath,koustuv.dasgupta}@xerox.com

Abstract. Services organizations are always under pressure to operate under tight costs and to improve their operational efficiency. Transaction processing is one of the major operations in a services organization. An organization is typically trained to serve a standard set of processes within different domains across several clients. Although each client has their own specifics with respect to a process, there is a lot of commonality within similar processes across clients. An organization's operational KPIs (i.e., Key Performance Indicators like processing time) across these clients when dealing with such related processes might not be similar; an organization might perform well for some clients and perform below par on others. There is a need to gain insights for such variance in performance and seek opportunities to learn from well performing client engagements (e.g., establish best practices) and leverage these learnings/insights on non-performing clients. We present a framework for analysing operational event data of related processes across different clients to gain insights on process executions. We present results of analyzing real-world transaction processing operations of a large services organization using the proposed framework. Our analysis shows that resource workload, clarity of process definitions, experience, and skill proficiency are key factors that influence the average processing time of transactions.

1 Introduction

Services organizations cater to a large number of clients on a daily basis. Each client specifies certain Service Level Agreements (SLAs) to quantify performance, e.g., turnaround time, cost, quality etc., which need to be met by service providers. Service providers typically implement/deploy a service delivery framework to meet these SLAs. Most service providers are challenged by a constant need to closely monitor the performance and efficiency of their operations to meet stringent compliance requirements, handle cost pressures, inefficient processes and complex workflows. Inability to meet SLAs due to inefficient business processes can amount to 20% loss for businesses today.

Many of today's service organizations record event data pertaining to their operations, which have been exploited to gain insights on process executions, for example, using process mining [1] techniques. Traditionally, such insights are obtained in silos, i.e., processes pertaining to clients are analyzed independently. In recent years, there has been considerable interest in cross-clientele analysis. The main motivation for such an analysis stems from the fact that - although service providers may be performing variants of a similar process across its clients, the Key Performance Indicators (KPIs)¹ may vary significantly across clients. For instance, two clients requiring a similar process (e.g., document verification) to be executed, may incur very different turnaround times for process completion. In some cases, this can even result in the service provider meeting SLA specifications for some clients, and violating those of others, despite the similarity in processes executed for the clients. This variation in business process performance can be attributed to variations in workflow design, resource allocation, context dependencies, skill deficiencies, etc. Given that services organizations frequently encounter the need to execute similar processes across its clients, a siloed approach of operational execution data analysis does not suffice. There is a need to gain insights for such variance in performance and seek opportunities to learn from well performing client engagements (e.g., establish best practices) and leverage these learnings/insights on non-performing clients.

In this paper, we propose an extensible framework using process mining that supports cross-clientele learnings based on analysis of event logs containing business events of various clients and their processes. The framework comprises of

- an analytic engine that provides insights on various aspects of a process such as process discovery, conformance checking (e.g., deviations from expected behaviour), and performance analysis (e.g., turnaround times, working times, bottlenecks)
- a root cause diagnostic engine that assists in identifying the potential causes of some observed behaviour
- a recommendation engine that provides prescriptions for improving the operations of an organization.

We present some insights from analyzing the operational data of transaction processing pertaining to a large services organization. In particular, we demonstrate how the proposed framework can elicit factors that influence the performance of key client processes.

The remainder of the paper is organized as follows. Related work is discussed in Sect. 2. Section 3 provides some background on process mining and event logs. The framework for cross-clientele analysis is presented in Sect. 4. Section 5 presents and discusses the application of the proposed framework on a real-life case study. Finally, Sect. 6 concludes the paper.

¹ Note that SLAs typically are external metrics (for clients) while KPIs are internal metrics (for the organization).

2 Related Work

While most prior works in process mining adopt a client-specific view, recent works have recognized the benefits of process analysis across clients [5, 6]. Buijs et al. [5] cross-correlate process models of different clients with respect to their actual behaviours observed from event logs along process model quality metrics (e.g., process complexity), behavioral quality metrics (e.g., throughput time) and comparison metrics (e.g., process similarity). However, they do not provide any root cause analysis of the observed behavior. Process-related risks pertaining to deadline transgressions or overruns are inferred in [12] using statistical techniques by identifying potential risk indicators such as abnormal activity execution/waiting time, multiple executions of activity etc. Root cause analysis of deviant process instances are studied in [8, 11, 15]. The basic idea in all of these is to define some features (e.g., workload, sequence of activities, etc.) and translate the problem of finding root-causes into a classification problem.

The impact of factors such as the complexity of work, priority or importance of work and expertise of the worker, on the operational productivity of the worker have also been studied [14]. However, [8, 11, 12, 14, 15] all focus on analyzing a single client/process. In contrast, our work analyzes several clients/processes and in addition, we explore the influence of factors such as resource skills/capabilities, their proficiency, team composition and process complexity on various behavioral aspects of process executions. Our work presents a holistic, data-driven framework with pluggable components which analyzes, infers and identifies opportunities for process improvements based on cross-learnings from event log data pertaining to various client process executions.

3 Background

In this section, we provide some background on process mining. Process mining serves as a bridge between data mining and business process modeling and analysis. The starting point of process mining is the notion of an event log. An event log captures the manifestation of events pertaining to the instances of a single process. A *process instance* is also referred to as a *case*. Each event e in the log corresponds to a single case and can be related to an *activity* or a *task* (for instance, e corresponds to audit task). Events within a case need to be *ordered*. An event may also carry optional additional information like *timestamp* (e occurred on April 2nd 2015 at 13:25:10 IST), *resource* (e was executed by Kathy), *transaction type* (e is a start event, i.e., the start of the auditing task by Kathy), and various data elements such as *costs* (e costed \$0.5), etc. Timing information of when an event occurred is required to analyze the performance related aspects (e.g., waiting time, processing time, etc.) of the process. Resource information such as the person executing the activity is useful when analyzing the organizational perspective, and in conjunction with time, can help in analyzing the productivity of various resources. We refer to these additional properties as *attributes*. To summarize,

- an event log captures the execution of a process.
- an event log contains process instances or cases.
- each case consists of an ordered list of events.
- each event can be associated exactly to a single case.
- events can have attributes such as activity, time, resource, etc.

Event logs from different systems and organizations can be stored in different formats, e.g., databases, plain text files, etc. For process mining applications, a common event log format based on a process meta model, called the MXML format (**M**ining **X**ML) [7], has been proposed. This has been followed by a more recent advancement, called the XES (**eX**tensible **E**vent **S**tream) [9]. XES is adopted as the standard by the IEEE Task Force on Process Mining.

A process should be analyzed holistically across four dimensions: (i) control-flow, (ii) data, (iii) resource, and (iv) time. Note that these dimensions are not orthogonal. The topics in process mining can be broadly classified into three categories (i) *discovery*, (ii) *conformance*, and (iii) *enhancement*.

- *Discovery*: Process discovery deals with the discovery of models from event logs. These models may describe control-flow, organizational aspects, time aspects, etc. There are dozens of process discovery techniques generating process models using different notations (Petri nets, EPCs, BPMN, heuristic nets, etc.) [1].
- *Conformance*: Conformance or compliance checking deals with comparing an apriori model with the observed behavior as recorded in the log and aims at detecting inconsistencies/deviations between a process model and its corresponding execution log. In other words, it checks for any violation between *what was expected to happen* and *what actually happened*. The apriori models can be specified using different notions such as Petri nets, Declarative models, or in the form of business rules.
- *Enhancement*: Enhancement deals with extending or improving an existing model based on information about the process execution in an event log. For example, annotating a process model with performance data to show bottlenecks, throughput times, etc., by exploiting the timestamps in the event log.

For a classic introduction to process mining, the reader is referred to [1]. We adopt several techniques from process mining and the process mining tool ProM [16]² to discover insights on process executions. While ProM supports analysis of one event log pertaining to a process, we built a plug-in in ProM that analyzes a collection of logs together. In addition, we built a root cause diagnostic engine to analyze the causes for observed behavior.

4 Framework for Cross-Clientele Analysis

In this section, we discuss an extensible framework for performing cross-clientele analysis of operations within a services organization. Figure 1 depicts the framework for cross-clientele analysis. The basic building blocks of the framework are discussed below:

² See www.processmining.org for more information and to download ProM.

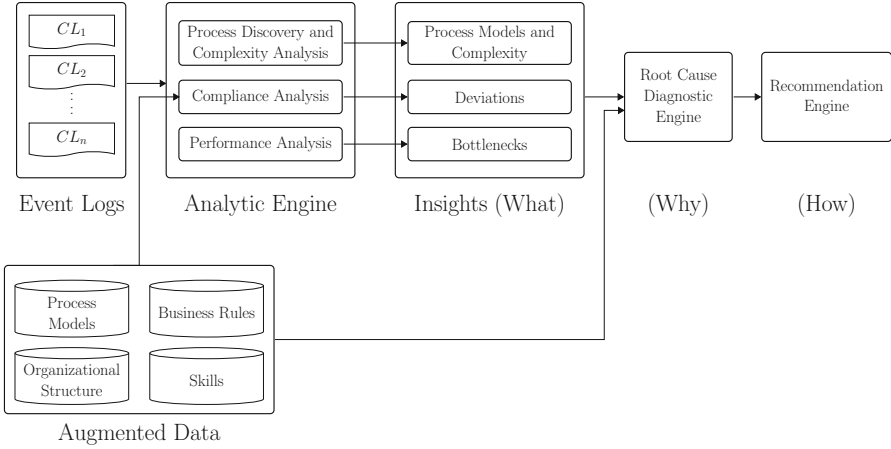


Fig. 1. Framework for analysis of process executions across clients.

- **Event Logs:** At the outset, we need to consider event logs pertaining to the operations of a particular process for all clients we are interested in analyzing. The particular choice of clients and processes can be driven from various means, e.g., one can choose clients across different geographies, based on their type such as gold, silver, and platinum, based on their KPI metrics (clients whose KPIs are met vs. those that are not), or can even be random.
- **Augmented Data:** In addition to the event logs, one can have other forms of data that augment the event logs. For example, one can have a repository of client process models corresponding to how the organization expects the process executions to happen, organizational structure eliciting the resources, their roles, departments, groups etc., data about the skills and proficiency of resources, business rules, etc. Some example business rules include: activity X should be executed by a resource with a proficiency at least 2 in skill S, activity Y should be executed within 2 h of executing activity X.
- **Analytic Engine:** Given a set of event logs and optional augmented data as mentioned above, one can derive several kinds of insights related to the process executions. The analysis techniques can be broadly divided into three categories:
 - *Process Discovery and Complexity Analysis:* This corresponds to the discovery of process models from event logs and analyzing them. We can compare how the process models vary across the clients, identify what is common between them, which workflow patterns are used in the processes, which client processes are complex (e.g., structural complexity), etc.
 - *Compliance Analysis:* This corresponds to verifying how compliant are client process executions with respect to their expected behavior. This takes as input event logs (capturing process executions) and process models or business rules (capturing expected behavior). One can apply replay techniques (replaying process instance traces onto the process models/

business rules) to identify what sort of deviations manifest, how often do they manifest in the process executions. If all client processes follow a similar process model, this gives a snapshot view of how compliant various client executions are w.r.t the expected behavior.

- **Performance Analysis:** This corresponds to various performance related aspects pertaining to the process executions, e.g., the average working/processing time of activities, the waiting times, sojourn times between activities, the turnaround time of the process etc. In addition, we perform resource performance analysis in the process, i.e., how do different resources execute an activity: the working/processing times that different resources take to execute an activity, deviations in execution times, etc.

Note that as mentioned above, insights should ideally cater to all the four perspectives (control-flow, data, resource, and time) of a process.

- **Root Cause Diagnostic Engine:** The insights that are uncovered in the earlier steps provide descriptive information as to what is happening in the operations of the process. Having discovered any interesting insights, the organization would be interested in identifying what the underlying (root) causes are for the observed behavior. The root cause diagnoser assists in identifying and corroborating factors that can be attributed to an observed behavior. The factors that influence the behavior of a process, e.g., resource skills and proficiency, process complexity, etc., can be (pre)defined. From the event logs and other augmented data sources, we extract the values corresponding to these features and the problem of finding the root causes can be posed as a learning problem. For example, for a given process, if we are interested in dissecting whether resource skill levels attribute to the variations in performance (working time), we can create a resource-skills matrix with each cell (i,j) in the matrix capturing the proficiency level of resource ‘i’ in skill ‘j’. We can classify the resources into various classes, e.g., efficient and poor, based on their performance. We can then apply techniques like decision trees to learn the discriminatory features if any between the defined classes. Simple correlation analysis techniques can also reveal any potential causes.
- **Recommendation Engine:** This corresponds to providing prescriptions to the organization towards improving their operations. The recommendations are driven by the root causes discovered in the earlier step, e.g., if skill proficiency is identified as a cause, one can recommend training of resources to improve their skill levels.

Note that the proposed framework can be extended with additional techniques. In this paper, we consider only the analytic and root cause diagnostic engines. The recommendation engine is beyond the scope of this paper. We are in discussions with the business stakeholders to translate/leverage some of our root causes to recommendations/prescriptions.

5 Cross-Clientele Analysis for Transaction Based Outsourcing (TBO) Business

In this section, we discuss the results of applying the proposed framework on the event logs pertaining to a transaction processing business organization within Xerox Services. The proposed framework has been implemented as a plug-in in the process mining tool ProM [16].

5.1 Business Context

For confidentiality, we anonymize the discussion on the business without losing any information. The organization serves clients across different domains $\mathcal{D} = \{D_1, D_2, \dots, D_n\}$. Within each domain, D , there are several processes $\mathcal{P}^D = \{P_1, P_2, \dots, P_m\}$. For each process, P , the organization has defined a set of steps that caters to the process. For a particular client C and a process in a domain, P^C , the organization configures the steps to cater to the needs of the client. These configurations result in different transaction types $\mathcal{T}_C^P = \{T_1, T_2, \dots, T_k\}$ for the client process.

The resources in the organization are skilled on the defined steps of a process. Different resources are skilled to execute different processes and the proficiency levels of resources on the skills can vary.

The organization has around 1400 employees and provides services across 4 primary domains and 163 processes for over 90 clients. The organization typically handles between 5000 and 10000 transactions per day. Although the daily volume of transactions is relatively moderate, we look at three months of longitudinal data to demonstrate the efficacy of our framework. We believe that the proposed framework is capable of handling scenarios with much larger volumes of transactions.



Fig. 2. An expected process behavior for the transactions.

The organization receives instances of a process as transactions. Each transaction refers to a process instance and the typical workflow of a process corresponds to the transaction being assigned by a “team lead” to a resource whose role is termed “processor”. The processor then works on the transaction (based on a well defined set of steps). The processor can either finish the transaction in one go or can process it for some time, suspend it and later resume it for completion. Once the processor completes his/her task, the transaction is then assigned to an “auditor” for quality check. The auditor checks if the processor has processed the transaction correctly and if no issues are perceived, the transaction is completed. If issues are found, the auditor sends the transaction back to the processor, who has to rectify the errors identified. Just like the processor, an auditor can perform his/her task in one go or can do the task for some

time, suspend it and later resume it for completion. The organization expects the transactions failing the auditor checks to be a rare event. Figure 2 depicts the process behavior for transaction processing.

5.2 Data Set

We have selected four processes that are commonly executed by five clients in domain A .³ These are some of the most critical processes that the organization provides services on and the chosen clients are among the big ones. The organization is interested in gaining insights on how these processes are managed across these clients. Let these processes be denoted by P_1^A , P_2^A , P_3^A , and P_4^A and the clients by C_1^A , C_2^A , C_3^A , C_4^A , and C_5^A . For these client-process combinations we have considered all transactions that have started and completed over a three month period between January 2015 and March 2015.

Table 1 depicts the number of transaction types and the average number of steps per transaction type for each of the client-process combinations. Also depicted in Table 1 is the minimum and maximum number of steps among the transaction types for a client-process combination. We can see that clients C_1^A and C_2^A have a lot of heterogeneity in their processes (this is reflected in the large number of transaction types for processes P_1^A , P_3^A , and P_4^A). At the same time, several of these transaction types are simple (involving only few steps), which is reflected in the relatively low average number of steps for these processes when compared to the other clients. The organization is particularly interested in the way how processors are executing their task. Hence in the rest of the discussion, we focus on insights related to processors. Table 2 depicts the volume of transactions, the number of “processors” who worked on these transactions, and the average number of transactions per resource for each of the client-process combinations. Also highlighted in the table are some aberrations. For example, we can see that client C_4^A exhibits drastically different characteristics when compared to others, the average number of transactions per resource is much larger (278.82) when compared to other clients for P_1^A and much lesser (4.22) for P_3^A . Similarly, for client C_5^A , the average number of transactions per resource is less (6.75) for P_3^A and much larger (429.57) for P_4^A .

For each of these transactions, the organization records event data in the form of a worklog. The worklog contains information about the high level steps of the process elicited above, e.g., when a transaction arrived, the team lead who assigned the transaction to a processor, when it was assigned, and to whom it was assigned etc. We process this data and generate XES/MXML event logs amenable for process mining. The next few sections provide a detailed analysis using the proposed framework in Fig. 1

5.3 Process Discovery and Complexity Analysis

As illustrated in the framework in Fig. 1, several types of insights can be obtained using various analysis techniques. Figure 3 depicts the process model discovered

³ Recall that the organization serves clients across different domains.

Table 1. The number of transaction types (#TT), the average number of steps per transaction type, the minimum, and the maximum number of steps among the transaction types for different processes across different clients in domain A.

	P_1^A				P_2^A				P_3^A				P_4^A			
	#TT	avg.	min.	max.	#TT	avg.	min.	max.	#TT	avg.	min.	max.	#TT	avg.	min.	max.
C_1^A	88	1.5	1	11	2	4.5	1	8	11	1.45	1	6	34	2.5	1	9
C_2^A	53	5.0	1	29	5	32.4	4	53	22	1.9	1	8	9	3.0	1	6
C_3^A	13	5.2	1	15	4	5.75	1	9	5	6.2	2	11	5	6.0	4	8
C_4^A	7	5.0	1	11	1	1.0	1	1	3	1.3	1	2	-	-	-	-
C_5^A	9	8.2	6	12	2	8.5	7	10	7	2.4	1	6	2	6.5	6	7

Table 2. The number of transactions (#T), the number of processors (#R) deployed, and the average number of transactions per processor for different processes across different clients in domain A.

	P_1^A			P_2^A			P_3^A			P_4^A		
	#T	#R	avg.	#T	#R	avg.	#T	#R	avg.	#T	#R	avg.
C_1^A	2917	19	153.52	1323	9	147	599	22	27.22	1804	15	120.66
C_2^A	7146	36	198.5	600	23	26.09	404	20	20.2	3052	38	80.31
C_3^A	1736	12	144.66	440	16	27.5	965	21	45.95	1572	20	78.6
C_4^A	6413	23	278.82	38	9	4.22	201	7	28.71	-	-	-
C_5^A	1494	8	186.75	54	8	6.75	444	14	31.75	6014	14	429.57

using the heuristics miner [17] for client C_3^A and process P_1^A . We can see that the workflow followed is along the expected process behavior illustrated in Fig. 2. In addition to the control-flow, the discovered process model provides insights about how often a particular activity and flow was executed. We can see that for roughly 3.8% of the transactions, processors have suspended and resumed the processing of the transaction. Similarly for 1.8% of the transactions, an auditor has suspended and resumed the auditing of the transaction.

Figure 4 depicts the process model discovered using the heuristics miner for client C_4^A and process P_1^A . In this model, we can see that for 72 transactions, an auditor has found a quality issue with how the transaction is processed by a processor. For a significant number of transactions 16% (unlike the earlier case where it was just 3.8%), the processor has suspended and resumed the transaction. Furthermore, the processors have (re-)suspended the transaction several times (263 instances) before they finally complete it. This clearly indicates that the processors had difficulty in executing this transaction (as is evident both from the number of times quality issues have been discovered and the number of times processors have to suspend and resume the transactions). *One reason for this could be attributed to the relatively large volume of transactions that a resource is assigned on average.* Recall from Table 2 that for this client-process, the average number of transactions per resource (278.82) is much larger than

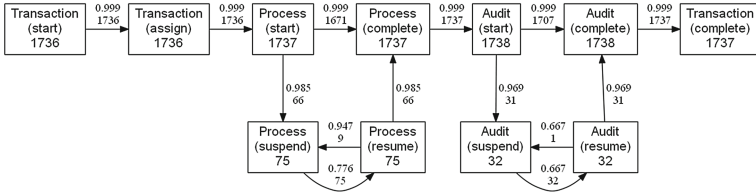


Fig. 3. Process model discovered using heuristics miner for client C_3^A and process P_1^A .

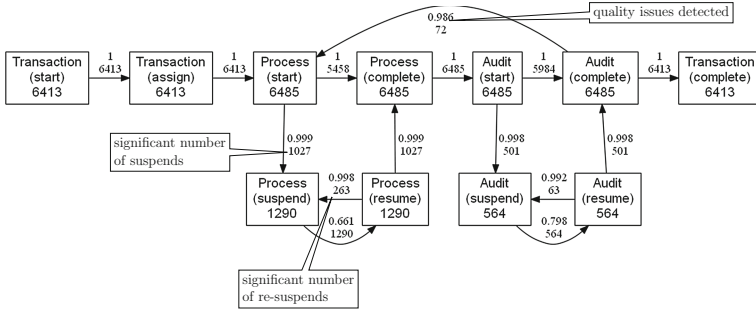


Fig. 4. Process model discovered using heuristics miner for client C_4^A and process P_1^A .

the average for this process across other clients. Process discovery techniques of our framework have thus been able to uncover the operational process model, even while providing insights w.r.t. the variants in control-flow aspects across client-process combinations.

5.4 Compliance Analysis

We next discuss some results on conformance checking (or compliance) analysis of the event data. Recall that conformance checking analyzes how conformant are real executions with respect to some expected behavior. One can specify the expected behavior in the form of a process model and replay techniques such as [2, 4, 13] can be applied. These techniques provide an objective *fitness* metric indicating how conformant a process instance is with respect to the expected behavior. In addition, they also reveal what deviations from the expected behavior are manifested in the process instance. One can also specify the expected behavior in the form of rules, for example as linear temporal logic (LTL) constraints for the control-flow perspective of a process and techniques such as [3, 10] can be used to identify the conformance or non-conformance of process instances. We have used the Petri net equivalent of the process model depicted in Fig. 2 as the model for expected behavior and replayed the event logs of each client-process using [4]. Table 3 depicts the results of conformance analysis where we elicit various classes of deviations uncovered, the client-processes, and the number of transactions where those deviations are manifested.

Table 3. Deviations uncovered during conformance checking analysis for the various client-processes of domain A.

Deviation	Client, Process	#Trans.
Transaction is assigned to two different processors	(C_2^A, P_2^A)	1
	(C_4^A, P_3^A)	1
Auditing is done by two different resources	(C_1^A, P_1^A)	1
Auditing is done after a transaction is completed	(C_3^A, P_1^A)	1
	(C_3^A, P_2^A)	5
	(C_3^A, P_4^A)	1
Quality issues are identified and processing is done more than once	(C_3^A, P_1^A)	1
	(C_4^A, P_1^A)	72
Transaction is withdrawn after processor completes the task	(C_3^A, P_2^A)	1

There are two instances where a transaction is assigned to two different processors. In one instance, (C_2^A, P_2^A) , the team lead first assigned the transaction to a processor who does not respond by accepting the transaction. The team lead then reassigns the transaction to a different processor. The other instance, (C_4^A, P_3^A) , presents an interesting scenario where the transaction is assigned to a different processor after the initially assigned processor has started working on the transaction. Deviations are observed in the audit executions as well. In one instance, auditing is done by two different auditors. The first auditor started auditing and suspended the task. It was later resumed by another auditor who completed the task. There are several instances where auditing was done after the transaction was completed. It is interesting to note that this deviation emanates only from operations related to client C_3^A across three different processes.

Recall from our earlier discussion that the organization expects quality check failures to be a rare event. We encountered two client-process combinations, (C_3^A, P_1^A) and (C_4^A, P_1^A) , where quality issues are perceived. Only one transaction failed the quality check in the former while 72 transactions failed the check in the latter (also as illustrated in Fig. 4). It is interesting to note that the quality issues are perceived only in process P_1^A . In one instance, (C_3^A, P_2^A) , a transaction is withdrawn after the processor finished executing the task. The deviations thus uncovered could then be analyzed by team leads and steps could be taken to prevent such deviations in the future.

5.5 Performance Analysis

We next discuss the results of performance analysis. Table 4 depicts the average and standard deviation of the working time that a processor takes to complete his/her task for the various client-process combinations. *The large deviations can be attributed to the variations in the number of check items (steps) among*

the transaction types of a client-process and also to the resources working on the process. We can see that processors take relatively very long time to finish tasks for all processes pertaining to client C_4^A as highlighted in the table. Note that P_2^A is defined over only one step for C_4^A and P_3^A over one/two steps unlike other clients. This implies that the entire process is defined as one/two steps rather than splitting into several smaller steps. The processor working on the transaction has to remember all that has to be done for C_4^A for this transaction using only these one/two check items (steps), which is difficult. Performance analysis can help us uncover bottleneck processes/tasks.

5.6 Root Cause Analysis

We analyzed if the number of steps in which processes are defined can *potentially be a root cause* for the variation in performance. The premise is that the more the number of steps, the finer the detail to which the process is explained.

Table 4. The average and standard deviation of working times (in seconds) taken by processors for different processes across different clients in domain A.

	P_1^A	P_2^A	P_3^A	P_4^A
C_1^A	2827 ± 9270	433 ± 2392	9878 ± 19259	1260 ± 4960
C_2^A	1618 ± 9201	5522 ± 19308	7624 ± 13566	4271 ± 21538
C_3^A	2349 ± 12806	9473 ± 29293	17087 ± 28643	1375 ± 4627
C_4^A	5282 ± 29129	44291 ± 60309	58180 ± 131275	
C_5^A	3364 ± 12787	3279 ± 12733	40792 ± 56348	529 ± 3770

Figure 5 depicts the influence of the number of steps on the processor’s working time of the transaction for various processes across all clients. We have considered the maximum number of steps as well as the average number of steps defined in a process and studied if there are any significant correlations. *We can see that as the maximum number of steps increases, the average working time decreases.* For a particular process, among clients where the maximum number of steps is the same, the average working time is directly proportional to the average number of steps in the process. For example, in Fig. 5(a) (also see Table 1), although both C_1^A and C_4^A have the same number of maximum steps (11), their average working times are 2827 and 5282 respectively (see Table 4). It can be concluded that, C_4^A takes relatively longer time because the average number of steps for this client is 5 as against 1.5 for client C_1^A .

Note that causal relations/correlations only suggest plausible factors for an observed behaviour. Valid conclusions can only be drawn in conjunction with process owners, while considering complete operational context. We believe this will have to be an iterative process, i.e., we discover insights from the data, present it to the process owners, incorporate their feedback, and if required repeat the analysis. The ability of a completely automated system for drawing confident conclusions largely depends on the goodness/quality of event data.

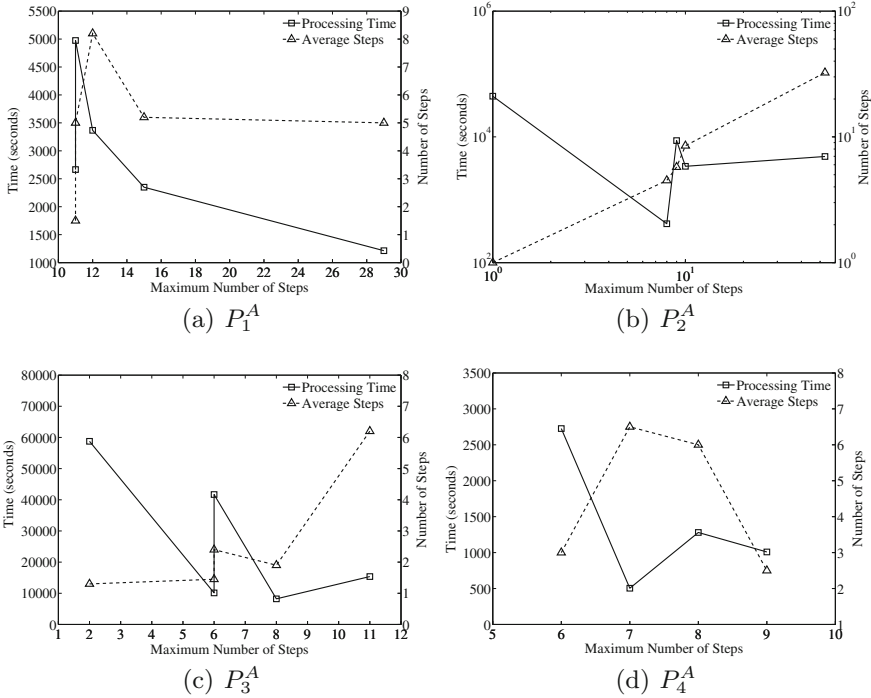


Fig. 5. Influence of the number of steps defined for a process on the processing time.

For example, apart from process data, one may also rely on information about resource activities each day, e.g., whether resources are working on other transactions with higher priority. Such completeness in event data logging is rarely observed. Therefore, for the study reported in the paper, we presented the uncovered causal relations to process owners, who concurred with the correlations, leading us to the stated conclusions.

In order to study the influence of resources on the variation in processing (working) time, we first filtered all infrequent transaction types and resources. We considered the transaction types that contribute to the top 95 percentile of transactions and resources who worked on at least 5% of transactions. Table 5 depicts the volume of transactions and the number of resources after this filtering. *We can see that the number of resources who regularly work on the process is much less when compared to all resources who worked on the process.* For example, for process P_1^A , there are 19 resources who worked on client C_1^A on all transactions but only 6 of them worked on at least 5% of transactions. When the volume of transactions arriving on certain days are high, in order to meet the SLAs, the organization deploys more resources to cater to the high volumes.

Figure 6 depicts the process model discovered using the heuristics miner for the client C_4^A and process P_1^A . We can see that in this model, the number of transactions with quality issues are 14 (0.35%) unlike the original model

Table 5. The number of transactions (#T), the number of processors (#R) deployed, and the average number of transactions per processor for different processes across different clients in domain A (considering only frequent transaction types and resources).

	P_1^A			P_2^A			P_3^A			P_4^A		
	#T	#R	avg.	#T	#R	avg.	#T	#R	avg.	#T	#R	avg.
C_1^A	1913	6	318.33	1189	2	594.50	504	11	45.82	1263	6	210.50
C_2^A	3711	5	742.20	328	7	46.86	252	6	42.00	987	6	164.50
C_3^A	1566	4	391.5	345	6	57.50	728	10	72.80	1109	6	184.83
C_4^A	4001	8	500.12	38	9	4.22	199	6	33.17			
C_5^A	1286	6	214.33	51	5	10.20	423	11	38.45	4617	5	923.40

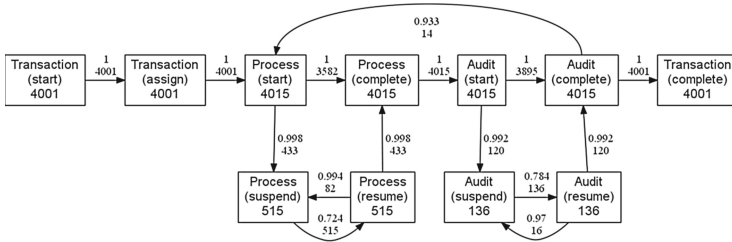


Fig. 6. Process model discovered using heuristics miner for client C_4^A and P_1^A considering only frequent transaction types and resources.

where there are 72 (1.13%) transactions with quality issues. *This indicates that resources who are deployed on a need basis being not so familiar with the process tend to do more mistakes.* Similarly, the number of times a processor suspends and resumes a transaction (10.82%) is lesser than the case when all resources are involved (16.01%).

Table 6 depicts the average and standard deviation of working time of the processors for the filtered log. We study if the experience and proficiency levels (in skills) of resources are potential causes for the variations in working times. Figure 7 depicts the relationship between the average working time of resources and their experience and skill proficiency for process P_1^A and two of the clients. We thus validate the intuitive results, that the *working times are inversely proportional to the experience of resources and directly proportional to their skill proficiency levels*, i.e., as resources gain experience, they tend to take less time. However, an experienced resource may take longer time than a less experienced resource if his/her proficiency level is lesser. We see an exceptional scenario in Fig. 7(b) where a resource with the maximum experience and proficiency level taking the longest time to execute this process. Upon further investigation, we uncovered that this resource, although processing large volumes of transactions for this process, is also involved in processing transactions related to another process. The context of the other process (demand, SLAs etc.) might require

Table 6. The average and standard deviation of working times (in seconds) taken by processors for different processes across different clients in domain A (considering only frequent transaction types and resources).

	P_1^A	P_2^A	P_3^A	P_4^A
C_1^A	2665 ± 9209	415 ± 2514	10125 ± 18704	1009 ± 2822
C_2^A	1213 ± 9015	4871 ± 18078	8214 ± 14503	2726 ± 15296
C_3^A	2349 ± 13268	8722 ± 27687	15373 ± 22996	1280 ± 4941
C_4^A	4974 ± 31383	44291 ± 60309	58760 ± 131807	
C_5^A	3367 ± 13333	3400 ± 13099	41692 ± 56969	504 ± 2236

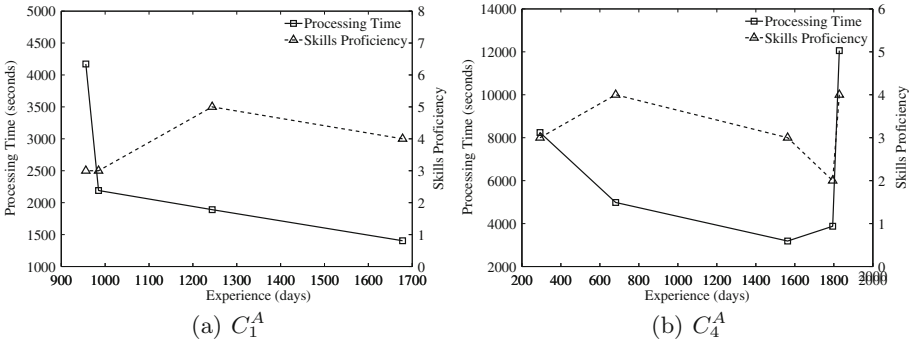


Fig. 7. Relationship between experience and proficiency in skills of resources and their working/processing time

him to be more involved in the execution of that process than this one, thereby resulting in longer working times for this process.

We have analyzed the event data pertaining to various client-process combinations in another domain and the basic observations discussed so far manifested also in the other domain. To summarize, our key findings are:

- the granularity at which process steps are defined influences the processing time; fine-granular steps help processors comprehend the task easily and thereby lead to efficient execution
- a resource’s experience and skill proficiency significantly impact the processing time
- adhoc resources deployed for executing a process to cater to large volume of transactions (on certain days) tend to make mistakes while executing the task (especially, for complex processes) impacting both the quality as well as processing time.

Based on our analysis, we prescribe that (i) processes should be defined in detail to help processors execute the process efficiently; (ii) stable resources should be deployed as far as possible to execute a process. Even if resources are to be

deployed temporarily to cater to the demand, it is advisable to have a limited pool of resources who are familiar with the process; (iii) resource experience and skill proficiency should both be considered when forming team compositions. These prescriptions are envisioned to be part of the recommendation engine after discussing with the business stakeholders.

6 Conclusions

Process improvement efforts based on insights obtained from the analysis of operational event data has primarily been applied on individual processes separately. Analysis of event logs pertaining to a similar process across different clients provides valuable insights on the operations of an organization. In this paper, we provided a framework that enables cross-client analysis and presented results on applying this framework on a real-life case study from a services organization. We uncovered that detailed description of process definitions, balanced workload, stable, proficient and experienced resources all lead towards operational excellence. Organizations can incorporate these learnings to aspects such as team formation (skill/experience dependent), training, workload distribution, task description etc. In addition to providing insights from cross-client learnings, we envision that our framework can assist service organizations in determining the health of client-process combinations and provide client-process benchmarks. As future work we would like to build a recommendation engine that provides prescriptions for operational excellence augmented with a simulation engine that corroborates with *what-if* analysis of the prescriptions.

References

1. van der Aalst, W.M.P.: Process Mining: Discovery Conformance and Enhancement of Business Processes. Springer New York Inc., New York (2011)
2. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisc. Rev. Data Min. Knowl. Discov.* **2**(2), 182–192 (2012)
3. van der Aalst, W.M.P., de Beer, H.T., van Dongen, B.F.: Process mining and verification of properties: an approach based on temporal logic. In: Meersman, R., Tari, Z. (eds.) OTM 2005. LNCS, vol. 3760, pp. 130–147. Springer, Heidelberg (2005)
4. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Conformance checking using cost-based fitness analysis. In: Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference (EDOC), pp. 55–64 (2011)
5. Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: Towards cross-organizational process mining in collections of process models and their executions. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part II. LNBP, vol. 100, pp. 2–13. Springer, Heidelberg (2012)
6. CoSeLoG (2014). http://www.win.tue.nl/coselog/wiki/project#the_project
7. van Dongen, B.F., van der Aalst, W.M.P.: A meta model for process mining data. In: Proceedings of the CAiSE Workshops (EMOI-INTEROP Workshop), vol. 2, pp. 309–320 (2005)

8. Ferreira, D.R., Vasilyev, E.: Using logical decision trees to discover the cause of process delays from event logs. *Comput. Ind.* **70**, 194–207 (2015)
9. Günther, C.W.: XES Standard Definition (2009). www.xes-standard.org
10. Maggi, F.M., Montali, M., Westergaard, M., van der Aalst, W.M.P.: Monitoring business constraints with linear temporal logic: an approach based on colored automata. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) *BPM 2011*. LNCS, vol. 6896, pp. 132–147. Springer, Heidelberg (2011)
11. Nguyen, H., Dumas, M., La Rosa, M., Maggi, F.M., Suriadi, S.: Mining business process deviance: a quest for accuracy. In: Meersman, R., Panetto, H., Dillon, T., Missikoff, M., Liu, L., Pastor, O., Cuzzocrea, A., Sellis, T. (eds.) *OTM 2014*. LNCS, vol. 8841, pp. 436–445. Springer, Heidelberg (2014)
12. Pika, A., van der Aalst, W.M.P., Fidge, C.J., ter Hofstede, A.H.M., Wynn, M.T.: Predicting deadline transgressions using event logs. In: La Rosa, M., Soffer, P. (eds.) *BPM Workshops 2012*. LNBIP, vol. 132, pp. 211–216. Springer, Heidelberg (2013)
13. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. *Inf. Syst.* **33**(1), 64–95 (2008)
14. Sindhgatta, R., Dasgupta, G.B., Ghose, A.: Analysis of operational data for expertise aware staffing. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) *BPM 2014*. LNCS, vol. 8659, pp. 317–332. Springer, Heidelberg (2014)
15. Suriadi, S., Ouyang, C., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Root cause analysis with enriched process logs. In: La Rosa, M., Soffer, P. (eds.) *BPM Workshops 2012*. LNBIP, vol. 132, pp. 174–186. Springer, Heidelberg (2013)
16. Verbeek, H.M.W., Buijs, J.C.A.M., Dongen, B.F.V., van der Aalst, W.M.P.: ProM 6: the process mining toolkit. In: *CEUR Workshop Proceedings*, vol. 615, pp. 34–39 (2010)
17. Weijters, A.J.M.M., van der Aalst, W.M.P.: Rediscovering workflow models from event-based data using little thumb. *Integr. Comput.-Aided Eng.* **10**(2), 151–162 (2003)

Pricing IT Services Deals: A More Agile Top-Down Approach

Aly Megahed¹(✉), Kugamoorthy Gajananan², Mari Abe², Shun Jiang¹, Mark Smith³, and Taiga Nakamura¹

¹ IBM Research – Almaden, San Jose, CA, USA
{aly.megahed, sjiang, taiga}@us.ibm.com

² IBM Research – Tokyo, Tokyo, Japan
{gajan, maria}@jp.ibm.com

³ IBM Global Technology Services, North Harbour, Portsmouth, Hampshire, UK
marksmith@uk.ibm.com

Abstract. Information technology service providers bid on high valued services deals in a competitive environment. To price these deals, the traditional bottom up approach is to prepare a complete solution, i.e., know the detailed services to be offered to the client, find the exact costs of these services, and then add a gross profit to reach the bidding price. This is a very time consuming and resource intensive process. There is a business need to get quick (agile) early estimates of both cost and price using a core set of high level data for the deal. In this paper, we develop a two-step top down approach for doing this. In the first step, we mine historical and market data to come up with estimates on the cost and price. We provide some numerical results based on industry data that statistically shows that there is a benefit of using historical data in this step beside the traditional way of using market data. Because the bidding price is not the sole factor affecting the chances of winning a deal, we then enter the different price points in a predictive analytics model (step two) to calculate the relative probability of winning the deal at each point. Such probabilities with the corresponding prices can provide significant insights to the business helping them reach quick reliable pricing.

Keywords: Service analytics · IT service deals · Predictive analytics · Pricing services · Estimating prices · Data mining

1 Introduction

Information Technology (IT) service providers compete to win high valued IT service contracts [1, 2]. Typically, clients ask for proposals that provide the pricing of particular services. Then, service providers prepare the deal pricing, submit their proposals, and enter a deal bidding process trying to win the contract.

The kind of services included in these deals are often complex, high valued, and very hard to quote [3]. Examples of these services include account management, storage systems, databases, and migrating the client infrastructure to the cloud.

Traditional practical approaches price deals via what we call the “bottom up” approach. This approach involves estimating the cost of individual activities at a granular level that together form a total cost for each individual IT service. Later, a markup/gross profit margin is added either to each service separately or to the sum of costs of all services in order to reach the overall price of the deal. Once this price is reached, usually solution designers assess the competitiveness of the solution by comparing it to historical deals and market data. Akkiraju et al. [3] provides a methodology for such assessment.

Note that services in this type of business are usually hierarchal. The first highest level of each service is always decomposed into smaller levels. For instance, end user is one of the common high level service that are usually further decomposed into hardware for end users, end user refresh (which refers to users who would get refresh/replacement for their assets),...etc. Hardware for end users can be further decomposed to different hardware devices,... and so on. For the bottom up approach to work, a detailed solution with all levels of all provided services needs to be prepared and costed. This can be time consuming and solution designers might not know all the detailed requirements at all service levels in early stages of the bidding process. Thus, there is a need to come up with an alternative “agile” approach that estimates prices of these complex IT deals with minimum information based on a typical scope of such services, e.g., using the highest level of services to be offered in the deal. “Agile” here refers to needing fewer inputs and coming up with prices very fast.

In this paper, we provide such an approach in a framework that consists of two parts. In the first part, we develop a calculation logic algorithm that comes up with the costing and pricing of the high level services included in a deal based on both market and historical data. While in the second part, we construct a methodology for predicting the relative win probabilities of these different price points. Using this “top-down” approach, solution designers can very quickly price complex deals and assess the relative chances of winning these deals for different pricing points. Figure 1 illustrates a comparison between the two approaches.

Additionally, service providers traditionally use market data collected from market consultancy companies to benchmark pricing generated from bottom up solutioning. In the first part of our framework, not only do we automate this process in an algorithmic fashion by estimating the cost and price directly using such market data, but we also statistically show that mining historical data in our top down approach might be more realistic in costing services compared to purely using market benchmark data. In opposed to market data, historical data is stored data of past deals for the same service provider.

Therefore, the contribution of this paper is three-fold. We develop an approach that both enables solution designers to determine price points of complex deals as well as assess the relative chances of winning these deals. Furthermore, we introduce the notion of assessing the cost of complex IT deals based on historical data and show that it could be effective for accurately costing some services compared to using the more traditional method of using market data.

The rest of this paper is organized as follows: in Sect. 2, we review the literature of related work. We present our approach with its two parts in Sect. 3, and then illustrate some results in Sect. 4. Section 5 ends the paper with the conclusions and some recommendations for future work.

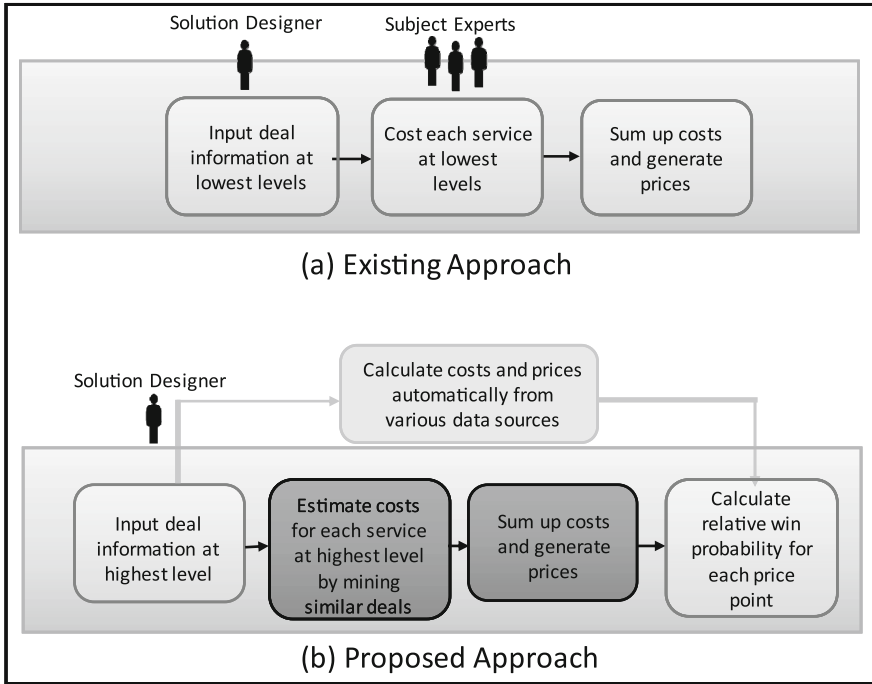


Fig. 1. Description of our proposed top down approach versus the existing bottom up approach

2 Related Work

The term “service” is often used to identify online services (e.g. web services) in the area of service-oriented computing. We note that “service” in this paper is broader than online services in the sense that our “services” identify and cover various constructs of IT services that include labor cost, management cost such as building customer relations and monitoring customer satisfactions, and other operational cost of human activities provided by IT solution vendors/providers. Gamma et al. [4] describe an Information Technology Infrastructure Library (ITIL) which provides a service design and catalog approach to organize service solutions. The service solutions that we study in this paper are organized using a particular taxonomy that follows the approach of our previous work in Akkiraju et al. [4] where one solution (deal) consists of a structure of particular hierarchical services. The top level is the highest level for the services and each service at that level is further decomposed into lower levels.

Pricing services has long been discussed in the literature of marketing and business management. Researchers have developed different pricing methods based on several pricing objectives. Avlonitis et al. [5, 6] summarized three categories of pricing methods linked with pricing objectives based on interviews of 170 service sector companies; (1) Cost based methods – where a profit margin is added on the cost, (2) Competition based methods – where pricing is done according to the market’s average prices or relevant to the competitors prices, and (3) Demand based pricing – in which the price is set to satisfy

customer's needs. Their results showed that the cost based method is the most adopted approach in their sample. However, it also showed that when the pricing objectives are associated with both the competitors and the customers, the possibility of adopting the market's associated pricing method increases. In [7], Tawalbeh stated an empirical study of a mobile service provider in which pricing was traditionally driven by a cost based method. The study concluded that service providers should focus more on market oriented pricing when the provider's pricing objectives are profit, market share, and sales maximization.

Multiple previous papers illustrated different methods of pricing some IT services. For instance, Basu et al. [8] developed a managerial guideline for pricing cloud offerings. Their approach models the utility of customers as a function of some parameters which have positive and negative effects on that utility. However, this paper (and its references) focuses on a particular service rather than the special case that we study here for deals composed of several complex IT services.

In the area of service sciences, several papers have been published investigating the assessment of "winnability" of IT services deals. The main relevant papers in this area are our previous works of Greenia et al. [1] and Megahed et al. [2]. In the former paper, the authors developed a predictive analytics model for predicting the winnability of in-flight deals, that is, deals that the service provider has already submitted the bid on. In the latter work, a similar predictive analytics model was developed but its focus was on predicting winnability of deals in an earlier stage, i.e., before submitting the bid. The main conclusion in both works is that the bidding price is not the only factor affecting the chances of winning IT services deals. Other attributes, such as competition, the type of client, and client's geographical location, have a statistically significance on the prediction of winnability. We incorporate these findings in the second part of our approach for pricing such IT deals. In the next Section, we develop our approach and explain its different pieces.

3 Methodology

As indicated in the introduction, our top down approach consists of two parts. In the first part, we provide a calculation logic that uses both historical data and market data in order to estimate service costs and prices of a deal using information about the highest level services included in such deal. Then, because the bidding price is not the only factor affecting the chances of winning a deal [1, 2], we adopt a predictive analytics model in the second part of our approach to come up with the relative probability of winning the deal corresponding to each price point. Below, we first start with some definitions and then we describe the two parts of our approach.

3.1 Definitions

We first differentiate between two categories of services that are included in any deal: (a) regular services (referred to as services below): for which cost will be independent of other services. Regular services are also services that have baselines/units. Examples of regular services are databases and end user, where the baseline for each is number of

databases and number of end users, respectively- and (b) common services: for which the cost is dependent on the different regular services included in the deal. An example for common services is account management, where its cost is determined depending on the costs/amounts of all regular services included in the deal since each of these services would need some account management.

Let D be the set of deals (historical and market data deals). We then define any deal $d \in D$ by the tuple of sets (*Meta Information*, *Services*, *Common Services*) where *Meta Information* is the set of the meta-data of the deal, namely:

$$\text{Meta Information} = \{\text{Deal Outcome}, \text{Contract Year}, \text{Geography}, \text{Industry}\}$$

where the *Deal Outcome* is either won or lost. Losing a deal might be due to another competitor winning it, it might be the case that the client decided to not pursue it anymore, or it might be because the service provider decided to not continue bidding on it. *Contract year* is the calendar year at which delivery of the services will begin. *Geography* and *industry* are the geographical location and industry of the client, respectively. The *Meta Information* can also be extended to include additional attributes of the deal. Remember that we are modeling this whole problem with respect to the service provider.

Services is the set of regular services, $\text{Services} = \{\text{Service}_1, \dots, \text{Service}_p, \dots, \text{Service}_M\}$, where M is the cardinality of the set *Services*. Similarly, *Common Services* is the set of common services, $\text{Common Services} = \{\text{Common Service}_1, \dots, \text{Common Service}_j, \dots, \text{Service}_N\}$, where N is the cardinality of the set *Common Services*.

Following the definitions of regular and common services, we define any regular service $\text{Service}_i \in \text{Services}$, where $i = \{1, \dots, M\}$, by the tuple (*Baseline*, *Cost*, *Price*). *Baseline* of a regular service Service_i is the unit/measure of the amount of the service provided by the IT service provider to a client. Refer to the two examples of databases and end user above. *Cost* is the total cost of Service_i and *Price* is the price of Service_i . Any common service $\text{CommonService}_j \in \text{CommonServices}$ is defined by the tuple (*Percentage of Total Cost*, *Cost*, *Price*), where *Percentage of Total Cost* is the cost of that common service as a percentage of the total deal cost, *Cost* is its total cost, and *Price* is its total price. Note that the total deal cost/price is the sum of the costs/price of all regular and common services. Also, note that the cost is what the service provider pays to provide the service (cost of labor, hardware, ... etc.), while the price is what is included in the bidding price, i.e., it is the cost with some profit margin (gross profit added to it).

Let us specify any scenario S as a new deal for which a solution designer needs to estimate its target price/cost. We assume that the following are given for such scenario: the elements of the sets Services_s , Common Services_s , the values of *Baselines* for each $\text{Service}_i \in \text{Services}_s$ and the elements of the set $\text{Meta Information}_s$, where each set is given the index s to specify that it is related to scenario S . Our target is to estimate the *Cost* and *Price* for each element of the sets Services and Common Services , and thus the total cost and price of Scenario_s . The next Subsection details our methodology of calculating these.

3.2 Peer Selection and Calculation Logic

Peer Selection. The first step in our calculation logic is to select peer deals. That is, we load historic and market benchmark data from the IT service provider's databases and carry out the deal selection at two stages. In the first stage, we use the *Meta Information_s* (*Deal Outcome, Contract Year, Geography, and Industry*) of the scenario to filter in the deals that have the matching values for respective fields. The reason behind this is that each of these fields is a characteristic of the deal and affects the cost of delivering each of its services. For instance, a service delivered from Asia is likely to have a different delivery cost compared to a service delivered to North America. Similarly, delivery a service in 2015 is likely to happen at a different cost compared to delivering the same service in 2016. Then, for each service in the *set Services* or *Common Services*, we filter out the peer deals that do not have that service. Thus, we have a different set of peers for each service of our scenario. The second stage of deal selection is to order these deals according to some criteria that we explain below.

Since there are two types of data sources - historic and market benchmark, peer selections are done separately for each source so that cost computation for a scenario's services can be computed in two perspectives. This is what we referred to above as the two different price points that we calculated using our approach.

We also ensure that a minimum required number peer historical deals for each service of a scenario exist in the database; if not, we report that no data is found for the historical deals prospective; so as not to report inaccurate results. A solution designer will specify the minimum threshold for the required number of peers, for each scenario. However, we do not specify that minimum number of peers for market data, as usually, there are a few market data/standard deal(s) for each service-Meta Data combination.

Sorting of Selected Peers for a regular service. \forall Service $i \in Services_s$, the sorting criteria of the set of peers selected for that service that we adopt is based on *baselines proximity*. $\forall i \in Services_s$, let *Baseline Proximity_{dsi}* be the baseline proximity between deal $d \in D$ and scenario S for service $i \in Services_s$. We define *Baseline Proximity_{dsi}* as follows:

$$\text{Baseline Proximity}_{dsi} = |\text{Baseline for Service}_i \text{ of deal } d \\ - \text{Baseline for service } i \text{ of scenario } s|$$

That is we assume that a deal and scenario are similar with respect to a service if the difference between the baselines is small. This assumption is justified because unit costs (which we will use below from peers to calculate the costs of our scenario) are typically similar for deals with similar/close proximity. That is because baselines define the complexity of the service and the variation of the unit costs for the same service across different deals is related to the quantity (baselines) of that service in each deal. The reason behind this is that service providers can usually achieve some kind of a quantity discount on unit costs for larger quantities. There is no set function that relates such quantity discount to the baselines and thus we adopt the baseline proximity to account for all that. Therefore, the outcome of the deal selection at service level is a set of similar deals, which are ordered based on their proximity value.

Sorting of Selected Peers for a common service. We sort peer deals for common services according to a different proximity. Let that proximity by *Common Service Proximity*_{dsj} (the proximity between deal $d \in D$ and scenario S for common service $j \in CommonServices$). Since common services do not have baselines and since they are related to the overall cost of regular services, we base that proximity on the total cost of regular services. That is:

$$Common\ Service\ Proximity_{dsj} = |Sum\ of\ Costs\ of\ regular\ services\ for\ deal\ d - Sum\ of\ costs\ of\ regular\ services\ for\ our\ scenario\ s|$$

We note that in order for us to calculate the above proximity, we first have to have calculated the costs for regular services of our scenario, which we show in the next calculation step.

Lastly, we set a maximum threshold T on the set of peer deals. Typically for market data, the threshold is 1, while for historical data, this can be set by the solution designer. We then use the top T peers in each ordered set of peers for each service to do our calculations below.

Calculation Logic. We here show how we calculate the costs for each service in the two sets of regular and common services for both the historical data and market data prospectives. Note that the cost calculation for each service is performed for each year of the total number of contract years of a scenario.

Cost calculation for Regular Services of a Scenario. For each regular service $i \in Services_r$, we first compute the unit cost of that service in each of its peer deals by dividing the cost of that service by its baselines. Then, we retrieve the l^{th} Percentile of these unit costs. Typically, one would use the median, but the solution designer can choose any arbitrary value for l . The rationale behind using the percentile is to allow the user to adjust for the complexity of the service if not captured by the chosen peers, i.e., if the unit costs of the selected peers are too diverse, then the user can input a percentile that is related to the complexity of the service that he/she might know and that we cannot capture automatically/algorithmically. Let us call the resulting unit-cost for service i : Unit-cost_i

Let *Baseline_i* be the baseline of service i of our scenario. Therefore, the cost of the regular service $i \in Services_s$ for our scenario S , *Cost_{s,i}*, is computed as:

$$Cost_{s,i} = Unit-cost_i * Baseline_i$$

Cost Calculation for Common Services of a Scenario. Since the costs of common services are related to the costs of regular services, for each common service $j \in CommonServices_s$, *Cost_{s,j}* (which is the cost of service j for our scenario S) will be computed as follows:

For each service $j \in CommonServices_s$, we calculate the percentage of the cost of that service to the overall cost of the deal for each peer deal in the ordered list of peer deals for that service. Then, we again apply an arbitrary percentile to these percentages to get the percentage of that service to that the total cost of our scenario S . We call the resulting percentage value of a common service $j \in CommonServices_s$ *P_{s,j}*

Now, the total cost of all services in our scenario S is

$$SUM_{s,all} = \sum_{j \in CommonServices_s} Cost_{s,j} + SUM_{s,reg}$$

Where $SUM_{s,all}$ is the total cost of the scenario (sum of the costs for all services; both regular and common ones), $SUM_{s,reg}$ is the sum of the costs for the regular services. Now we have that for each $j \in CommonServices_s$ in our scenario S :

$$Cost_{s,j} = SUM_{s,all} * P_{s,j}$$

We thus transform the above set of linear equations to a standard format as:

$$\begin{aligned} (P_{s,1} - 1) * Cost_{s,1} + P_{s,1} * Cost_{s,2} + \dots + P_{s,1} * Cost_{s,J} &= - P_{s,1} * SUM_{s,reg} \\ P_{s,2} * Cost_{s,1} + (P_{s,2} - 1) * Cost_{s,2} + \dots + P_{s,2} * Cost_{s,J} &= - P_{s,2} * SUM_{s,reg} \\ &\dots \dots \dots \\ P_{s,j} * Cost_{s,1} + P_{s,j} * Cost_{s,2} + \dots + (P_{s,j} - 1) * Cost_{s,J} &= - P_{s,j} * SUM_{s,reg} \end{aligned}$$

Where J is the cardinality of the set $Common Services_s$. By using the Cramer’s rule [9], we solve the above equations to compute the cost of each common service per year.

Since the only difference in calculation steps between historical data and market data is that for market data we typically have a maximum of 1 (or a few) market deals, we do not apply the percentiles for calculating unit costs (for regular services) and unit percentages (for common services) for the market data calculations when that maximum threshold is 1. Other than that, everything else is exactly similar to historical data calculations.

In the Sect. 4, we show the usefulness of using historical data in addition to the more traditional adoption of market data through some numerical experiments.

Now, adding up the costs of both regular and common services, we reach the estimated cost of a deal, for each of the historical data and market data cases. Then, by adding a chosen arbitrary gross profits (GPs) to the cost, we get different price points. Our overall approach, as can be seen from the details in the previous subsections, uses a minimal amount of inputs from the user and generates prices very fast, and thus is “agile” as required by modern business practices in this type of industry.

To assess the relative chance of winning the deal corresponding to each price point, we use a win prediction model discussed in the next subsection.

3.3 Win Prediction

We use the predictive analytics model developed in our earlier work in [2]. The model is based on the well-known naïve Bayes classifier. We refer the reader to references [10, 11] for an explanation of the naïve Bayesian model. The factors included that were shown to be significant are some of the deal attributes, in addition to some derived parameters. Beside the bidding price, we summarize the other significant factors used in the model as follows:

Complexity of the Deal. Complexity is determined based on the number and effort of delivery of the offered services to the client.

Global Versus Local. Deals are global if the services will be delivered to multiple countries. Local deals are ones in which services are delivered to one or two countries of close proximity (e.g., Australia and New Zealand).

Key Services Delivery Executive. Deals are sometimes assigned to a delivery executive responsible for the delivery of services after contract signing. The parameter here is whether a delivery executive is assigned early on for the deal or not.

Third Party Advisor. A third party advisor is used by some clients. The parameter here is whether the client has such advisor or not.

Contract Length. The number of years of the deal delivery.

Client-Market Segmentation. Clients are classified based on size, market audience, and market potential.

Number of Competitors. This is a count of the number of other service providers competing to win the same deal.

Competitor Classification. Competitors are classified according to whether they provide cloud, software, and network, whether they are niche or consultant.

The model is fairly accurate. It produces an average accuracy of 86 % and 93 % on training and testing data, respectively. The idea here is that multiple copies of the deal that we are trying to price will be entered as testing data to the model. All copies share the same meta data/attributes, except for the bidding price. Each copy has a different price point, out of the ones we calculated above (as well as any user chosen price). Note also that since the GPs are arbitrary, multiple GPs can be applied to the calculated costs and more copies of the same deal can be added. The predictive model will then output a ranked list for these copies and provide a relative winning probability score for each price point. Using that way, we are able to quantitatively/analytically assess different bidding price points given the fact that price is not the only factor affecting winnability and we can thus get a chart like the one in Fig. 2. The chart shows different pricing options (cost + GP) with the corresponding relative winning probabilities. Figure 3 gives an overview of the architecture of our overall approach.

4 Numerical Results

In the bottom up approach, accurate costs of services in the IT deal are evaluated at the lowest levels and summed up to come up with the costs of services at the higher levels. In coming up with fast evaluations of the costs of services in the early stages of the bidding process, traditionally, solution designers use market data. In the first part of our approach described in the previous Section, we proposed mining historical deal data besides market data. In this Section, we conduct some experiments to show that doing

so might be beneficial, i.e., might result in costs that are closer to the more accurate actual costs obtained using the detailed traditional bottom top approach.

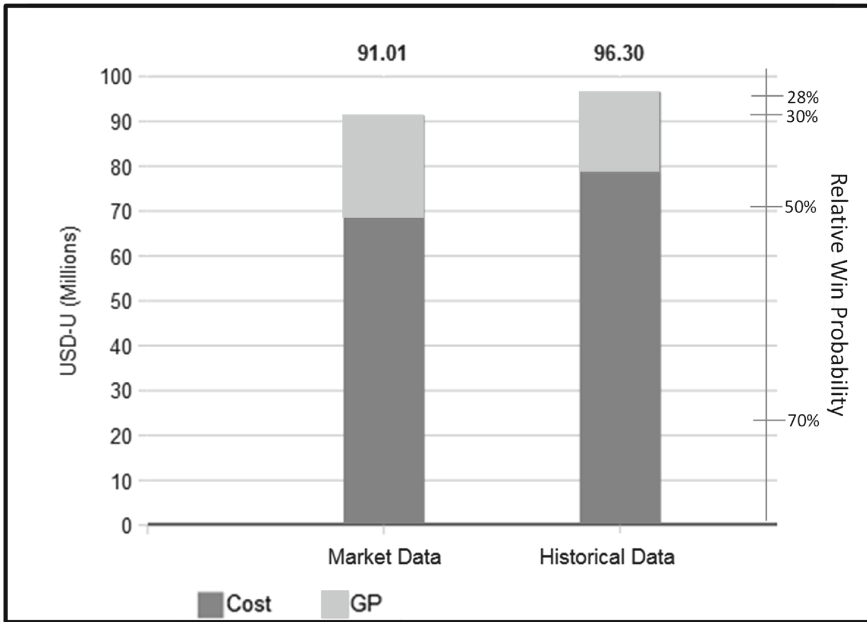


Fig. 2. Different price points with their corresponding values and relative winning probabilities

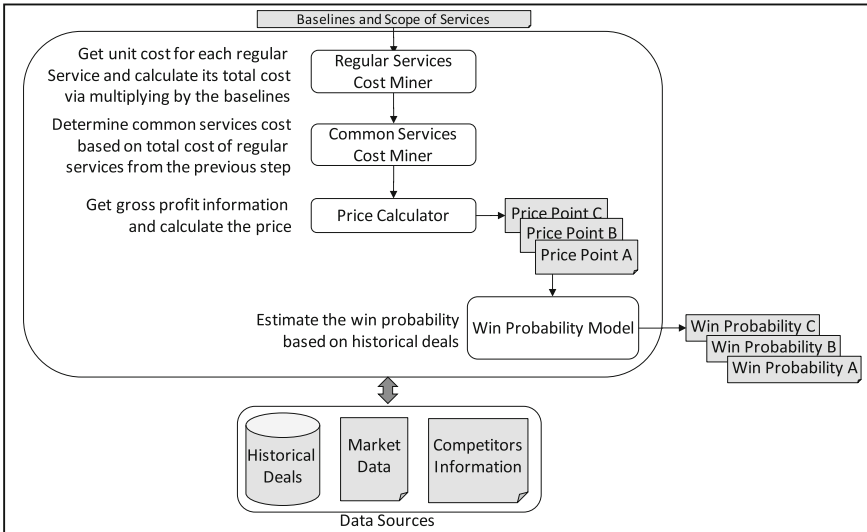


Fig. 3. Architecture/Overview of our overall approach

For our experiments, we selected 39 deals at random from a repository of real industry historical deals for an IT service provider that has complete costs cases. For each of the deals, we used baselines at the highest level for the services included in the deal and performed our calculation algorithm described in Sect. 3.1 to get market data and historical data costs. Then, we compared these costs with the actual costs in the sample data. The metric we used is the relative absolute difference between the calculated value and the actual value. Thus, for historical data and market data, respectively, the error would be:

$$Error_{historical\ data} = \frac{|Actual\ Cost - Cost\ Calculated\ Using\ Historical\ Data|}{Actual\ Cost}$$

$$Error_{Market\ data} = \frac{|Actual\ Cost - Cost\ Calculated\ Using\ Market\ Data|}{Actual\ Cost}$$

Note that we do the comparison at the cost level since prices are calculated by adding arbitrary GPs. Note also that the accuracy of cost estimation of a new deal does not imply a higher probability of winning the deal; since accurate costs do not imply competitive costs/prices and even competitive costs/prices are not the sole predictive factor for winnability, as discussed in the previous Section of this paper.

We calculated the error for each service out of 13 services for the 39 sample points for both market data and historical data. Then, for each service, we performed a *paired t-test* to test the following hypothesis:

$$H_0: \mu_D = 0$$

$$H_1: \mu_D < 0$$

Where,

$$\mu_D = \mu_{historical\ data\ error} - \mu_{market\ data\ error}$$

Here, μ_D is the difference between the mean of the historical data error (denoted as $\mu_{historical\ data\ error}$) and that of market data (denoted as $\mu_{market\ data\ error}$). For the used test of hypothesis, we used the same notation, assumptions, and details in the texts of Montgomery et al. [10] and Walpole et al. [11]. The test is justified since calculations of each of market data and that of historical data were done independently on each service using the same historical complete costs cases/deals. After assessing the assumptions of the test, we calculated the *p-value* for each service. Table 1 illustrates the results of the tests.

One can see from the results that there is a statistical evidence/significance that using historical data would yield more accurate costs than using market data for some of the services. This illustrates the usefulness of adopting the historical data mining into our approach. We next state the conclusions and directions for future work.

Table 1. *P-value* results for the paired t-test of each service

Service number	<i>P-Value</i>	Reject H_o at a significance level of 0.05	Reject H_o at a significance level of 0.1
1	0.210		
2	0.048	x	x
3	0.036	x	x
4	0.047	x	x
5	0.091		x
6	0.138		
7	0.068		x
8	0.044	x	x
9	0.003	x	x
10	0.044	x	x
11	0.266		
12	0.392		
13	0.065		x

5 Conclusion and Future Work

In this paper, we provided an approach that not only gives a quick agile estimate for the costs and prices of information technology complex services deals with minimal input, but also assesses the relative probabilities of winning such deals for each price estimate. Our approach consists of two phases. In the first phase, we used both historical and market data to estimate the costs. In addition, we showed experimental results based on industry data that illustrates that using historical data is more accurate in estimating the costs for many services, when compared to using market data (which is the more traditional business approach). In the second phase of our approach, we incorporated our price estimates in a predictive analytics model to come up with relative winning probabilities corresponding to each price point. Providing this output helps the solution designers and business executives decide on the final bidding price they would like to pursue.

There are several directions for future research to this work. Instead of estimating the costs/prices based on the highest level of the services, if the solution designer knows a little more detail (e.g., for the second highest level baselines of the offered services in the deal), then one can estimate the costs/prices based on historical and market data at that level. One challenge would be that not all chosen peer deals have all these services

at that second level in them. Thus, some machine learning approach might need to be used to compensate these values. Another direction for future research would then be comparing the accuracy of the cost estimation based on the top level of services (as we do in this paper) with that of the second level.

References

1. Greenia, B.D., Qiao, M., Akkiraju, R.: A win prediction model for IT outsourcing bids. In: Service Research and Innovation Institute Global Conference, pp. 39–42 (2014)
2. Megahed, A., Ren, G., Firth, M.: Modeling business insights into predictive analytics for the outcome of IT service contracts. In: Proceedings of the 12th IEEE International Conference on Services Computing (SCC), pp. 515–521 (2015)
3. Akkiraju, R., Smith, M., Greenia, D., Jiang, S., Nakamura, T., Mukherjee, D., Pusapaty, S.: On pricing complex IT service solutions. In: Service Research and Innovation Institute Global Conference, pp. 55–64 (2014)
4. Gamma, N., Do Mar Rosa, M., Da Silva, M.: IT services reference catalog. In: IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 764–767 (2013)
5. Avlonitis, J.G., Indounas, A.K.: Pricing objective and pricing methods in the service sector. *J. Serv. Mark.* **19**(1), 47–57 (2005)
6. Indounas, K., Avlonitis, G.J.: Pricing objectives and their antecedents in the services sector. *J. Serv. Manage.* **20**(3), 342–374 (2009)
7. Tawalbeh, M.: The impact of marketing-oriented pricing on product mix pricing strategies – an empirical study on the mobile telecommunication providers in Jordan. *Int. J. Econ., Commer. Manage.* **III**(1) (2015)
8. Basu, S., Chakraborty, S., Sharma, M.: Pricing cloud services—the impact of broadband quality. *Omega* **50**, 96–114 (2015)
9. Adhikari, M.R., Adhikari, A.: Text Book of Linear Algebra: Introduction to Modern Algebra. Allied publisher Pvt Ltd. (2005)
10. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics, 2nd edn. Springer, NY (2009)
11. Russell, S., Norvig, P.: Artificial Intelligence: A modern Approach, 3rd edn. Prentice Hall, PA (2009)
12. Montgomery, D., Runger, G.: Applied Statistics and Probability for Engineers, 5th edn. Wiley, New York (2010)
13. Walpole, R., Myers, R., Myers, S., Ye, K.: Probability and Statistics for Engineers and Scientists, 9th edn. Pearson, Boston (2011)

Demonstration Track Papers

SimMon: A Toolkit for Simulating Monitoring Mechanism in Cloud Computing Environments

Xinkui Zhao¹(✉), Jianwei Yin¹, Pengxiang Lin¹, Chen Zhi¹, Shichun Feng¹,
Hao Wu¹, and Zuoning Chen²

¹ College of Computer Science, Zhejiang University, Hangzhou, China
{zhaoxinkui,zjuyjw,pxlin,zjuzhichen}@zju.edu.cn

² National Parallel Computing Engineering Research Center, Beijing, China
chenzuoning@163.net

Abstract. Monitoring is significant to supervise the state of services and guide adaptive management of services in cloud computing environments. Working as auxiliary tools, monitoring systems are expected to incur the least extra cost on physical resources (CPU, memory, network, etc.). Since the scale and requirement of different data centers vary from each other, it is impossible to design a suit-to-all monitoring solution for all the data centers. However, for a certain data center, it is hard to determine whether a predesign monitoring mechanism is well suited before the mechanism is deployed in a real production environment. To address these issues, we propose SimMon, a toolkit for simulating monitoring mechanism in cloud computing environments. SimMon is used to simulate the process on collection, dissemination, storage and requisition of monitoring data. With the help of SimMon, system administrators are able to compare different monitoring mechanisms and select the best one before it is adopted by a monitoring system in a real-world data center.

1 Motivation

Fueled up by the explosive growth of services in cloud computing environments, traditional predesigned and suit-to-all monitoring tools are not efficient enough for cloud monitoring for three reasons. (1) The number of monitoring target becomes huge, which makes the traditional centralized management structure incapable to efficiently coordinate these dispersed collection agents. Efficient and distributed organization of the monitoring agents are required to ensure the performance of monitoring systems [2]. (2) The volume of data that are disseminated across data centers is large, which incurs much more extra network pressure [4]. To eliminate the extra pressure, tricky strategies, such as dynamic

This work was partially sponsored by National Natural Science Foundation of China under Grant (No. 61272129), National High-Tech Research Program of China (NO. 2013AA01A213), New-Century Excellent Talents Program by Ministry of Education of China (No. NCET-12-0491), Zhejiang Provincial Natural Science Foundation of China (No. LR13F020002), Science and technology Program of Zhejiang Province (No.2012C01037-1).

control on monitoring target, monitoring interval, data polling strategy, etc., are necessary to handle collected data [3]. (3) In cloud environments, services are located on infrastructures across different regions, which makes the underlying network structure for data dissemination complex. To ensure fast data access and high availability, new protocols that provide better solution for monitoring data storage and requisition are imperative [5].

Considering the above reasons, it is vital to design monitoring mechanisms according to the characteristics and monitoring requirements of a specific data center in cloud computing environments. In this work, we propose SimMon, a toolkit to simulate monitoring mechanisms and evaluate their effectiveness. SimMon is used in two main scenarios: (1) to test whether a monitoring strategy would work well in a certain data center before it is deployed and run in a real production environment; (2) to compare the results of different strategies and decide which strategy is the most appropriate one for a specific data center.

2 Architecture of SimMon

Figure 1 depicts the architecture of SimMon. It is composed by four main components: network, data storage, data dissemination and strategy control panel.

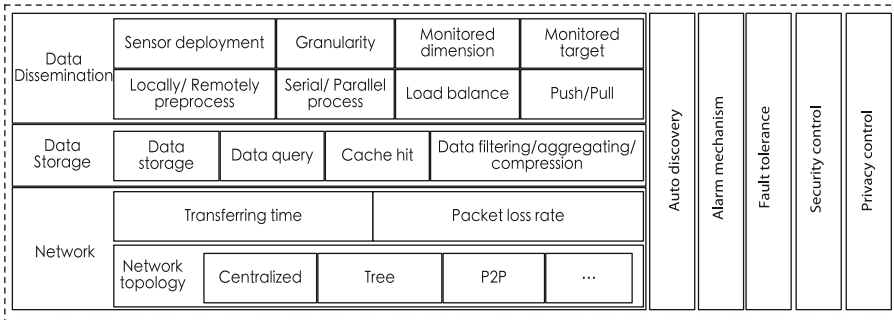


Fig. 1. Architecture of SimMon

Modeling Network. Network layer is an important consideration since the bandwidth and time costed by data transmission highly rely on underlying network structure and the logical topology of monitoring systems. In order to model the network structure, we simulate the behavior of root switch, aggregation switch and access switch separately and combine them as a layered structure. Apart from the underlying network structure, logical monitoring topology focuses on the organization of monitoring agents. A monitoring system commonly consists of three kinds of agent: collection agent (also called as sensor), federation agent and root agent. Collection agents are hosted on the same virtual machine with the service target to locally collect monitoring measurements. Federation agents are in charge of data organization and processing for a subset of collection agents. Root agents act as central nervous to control the global

scheduling strategies of monitoring systems. We define the three kinds of agents to support the design for centralized, tree-based, P2P-based, and hybrid topologies. We adopt an event-based mechanism to control the process on packets transformation and handle the packet loss situation. Latency between nodes is calculated from the underlying network structure and a BRITE-style file that contains delay metrics between each pair of virtual machines.

Modeling Data Storage. In monitoring systems, collected data are usually transferred from collection agents to federation agents and stored in a data repository for future query and analysis. To model the data storage process, we give an interface to simulate different data repositories, such as MySQL and HBase. Concurrently with the support for database simulation, the organization of the storage nodes in a distributed database is also important to reduce the total network bandwidth cost and the chance for resource conflicts. A good algorithm should consider the data volume to be transferred and the resource usage of business-related workload in the data center. Furthermore, in data query process, it is important to find the shortest route to get requested data. We implement a cache-hit strategy to store the data that collected in the most recent period in cache for fast query. To ensure high availability, we design a structure to support users to define different replication strategies. More than one copy of replication of the collected data are stored in replication servers in case of emergency.

Modeling Data Dissemination. In a distributed monitoring system, monitoring data are collected by collection agents and disseminated to federation agents. In the dissemination layer, there are three main processes that may cost extra resources: getting monitoring data, disseminating the data, and receiving the data. In the process of getting monitoring data, we simulate strategies to deploy bunches of sensors intelligently and implement algorithms on precise target selection, accurate collection interval selection and dynamic data preprocessing to reduce the data from source. In the process of disseminating the data, we reduce data dissemination actions by intelligent strategies on load balancing and data polling (data collected by the dispersed collection agents can be pushed to federation nodes passively or be pulled by federation nodes proactively). In the process of receiving the transferred data, we design two protocols: unicast protocol and multicast protocol. An unicast protocol can improve the accuracy of delivered data, while a broadcast protocol brings efficiency for data delivery.

Modeling Strategy Control Panel. Sensors are the source of monitoring data, and they are developed and deployed individually with monitoring systems. Meanwhile monitoring systems should be capable to discover independent sensors and add them into management consoles. There are two main solutions to discover newly installed sensors: event-based announcement from the installed sensors and periodic scan from federation agents or root agents. We implement security and privacy policies by creating subnets for a certain set of sensors. Monitoring systems are expected to send alarms to system administrators when a certain kind of event occurs. To filter out those false alarms, we build an interface to support users to redesign the alarm strategy and compare their results.

3 Implementation

On the design of SimMon, we first adopt the classes that are inherited from CloudSim [1] to build a testbed that contains hosts, switches, virtual machines, and workloads, and the testbed is a simulation of a data center in cloud computing environment. Based on the simulated data center, we develop a new toolkit to support users to build different monitoring mechanisms. We use Java language to implement the simulation toolkit and the toolkit program contains 15890 lines of code in total. Source code of SimMon is available at <http://www.cmsci.net/pxlin/simmon>.

4 Demonstration

In the demonstrations, we first use SimMon to simulate a cloud data center with 10000 physical servers (PSs), and each PS host 16 virtual machines. The PSs are dispersed in 10 individual small-scale data centers, and they are connected by a tree-based underlying network. Workloads running in the cloud environment are simulated with certain distributions. The simulated cloud data center is the target that we want to monitor. Hence all the monitoring mechanisms are designed based on the data center. We use three examples to demonstrate three common usage scenarios of SimMon.

- **Influence of different topologies in monitoring systems.** In this demonstration, we build four monitoring systems with different monitoring topologies: star-based, tree-based, P2P-based, and hybrid. In each monitoring system, we first simulate a data polling strategy that pushes collected data with certain interval to federation agents. We then summarize the total extra cost caused by monitoring systems and compare the influence of different topologies.
- **Data dissemination cost comparison by different polling strategies.** In the demonstration, we implement three data polling strategies: push at a certain interval, hybrid push and pull, intelligent exchange between push and pull. Based on SimMon, we compare the extra cost and accuracy of these strategies.
- **Effective alarm reduction by different alarm strategies.** In this demonstration, we test three different strategies on producing alarms: alarm on CPU usage, alarm on memory usage and alarm on CPU and memory usage. Based on SimMon, we compare the number of effective alarms that are caused by the three strategies.

References

1. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Experience* **41**(1), 23–50 (2011)

2. Jain, N., Kit, D., Mahajan, P., Yalagandula, P., Dahlin, M., Zhang, Y.: Star: self-tuning aggregation for scalable monitoring. In: Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB 2007, pp. 962–973. VLDB Endowment (2007)
3. Lu, X., Yin, J., Li, Y., Deng, S., Zhu, M.: An efficient data dissemination approach for cloud monitoring. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) ICSOC 2012. LNCS, vol. 7636, pp. 733–747. Springer, Heidelberg (2012)
4. Meng, S., Iyengar, A.K., Rouvellou, I.M., Liu, L.: Volley: violation likelihood based state monitoring for datacenters. In: Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems, ICDCS 2013, pp. 1–10. IEEE Computer Society, Washington, DC (2013)
5. Wang, C., Schwan, K., Talwar, V., Eisenhauer, G., Hu, L., Wolf, M.: A flexible architecture integrating monitoring and analytics for managing large-scale data centers. In: Proceedings of the 8th ACM International Conference on Autonomic Computing, pp. 141–150. ACM (2011)

CASE: A Platform for Crowdsourcing Based API Search

Tingting Liang^(✉), Liang Chen, Zhining Xie, Wei Yang, and Jian Wu

Zhejiang University, Hangzhou, China

{liangtt, cliang, lynntse, victor0118, wujian2000}@zju.edu.cn

Abstract. With the rapid growth of Web APIs on the Internet, searching appropriate APIs is becoming a challenging problem. General API search systems (e.g., ProgrammableWeb) implement API search through simple keywords matching leading to unsatisfactory search results. In this paper, we presents a crowdsourcing based API search engine CASE. Specifically, the API search engine leverages social information, *Twitter List*, a tool used by individual users to organize accounts that interest them on semantics. Based on the lists information, Latent Semantic Indexing (LSI) model is employed to compute the semantic similarity between the APIs and queries. Furthermore, the popularity of APIs inferred from the lists number is integrated with the semantic similarity to generate the final search result.

1 Introduction and Motivation

With the development of Web and mobile applications, a form of service called application programming interface (API) becomes prevalent. An API is a gateway to other people’s software and can be employed to invoke a third-party software component over the Internet. Compared to the traditional web services, APIs are in a more popular style and can make solution more accessible and more useful, which lead to a rapid growth of Web APIs. According to the analysis offered by ProgrammableWeb (PW)¹, a platform dedicated to manage APIs and mashups, the number of its following APIs increased fast in recent years and has reached 10850 by October 2014. Thus, how to discover the appropriate APIs becomes a hot issue. Since general API search systems consider the keywords match rather than the real semantic or topic information of APIs, the performance of API search is limited. Taking PW as an example, if a consumer takes “travel” as the query in search system, the top result is “[Webcams.travel](#)” which is a directory of touristic webcams and classified in *Video* category. Absolutely, “[Webcams.travel](#)” is not the objective API. The reason is that PW search system only seeks APIs whose names or descriptions contain the query.

To alleviate the limitation of general API search systems, introducing crowdsourcing information into the search method is a popular and novel strategy. And it has been proved effective in many other fields, such as information retrieval,

¹ ProgrammableWeb: <http://www.programmableweb.com>.

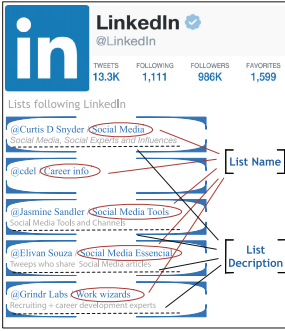


Fig. 1. An example of list

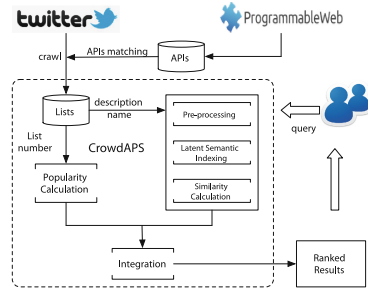


Fig. 2. Framework of crowdsourcing based API search

social network, etc. [2, 4]. In this paper, we introduce extra crowdsourcing information, named as Twitter lists [3], to improve the performance of API search. Twitter lists, which are used to help users organize the Twitter accounts they are interested in, include list names and descriptions that contain a wealth of semantic information. Figure 1 illustrates the contents of lists by showing several lists including LinkedIn. Most of the list names are related to ‘Social Media’ and ‘Career’ along with the corresponding list descriptions, which distinctly reflect the topic of LinkedIn. After being matched with APIs in PW, we crawled 3877 APIs’ Twitter accounts and the lists involving them. One API could be involved in many lists and the most reaches 84511. 30.8% of total APIs are organized by more than 100 lists, and 70.1% are included by at least 10 lists. Thus the list information of most APIs is full of richness.

In this paper, we build a demonstration of API search engine called CASE. The critical idea of CASE is to infer an API’s topic by analyzing semantics of lists including the API Twitter account. Based on the lists information, Latent Semantic Indexing (LSI) [1] is employed to compute the semantic similarity between APIs and queries. Additionally, the popularity of APIs inferred from the lists number is integrated with the semantic similarity to generate the final search result. Therefore, CASE could provide semantics matched and popularity satisfied results for a given query.

2 Algorithm Framework

Figure 2 illustrates the whole framework of algorithm applied in the API search engine. The list data for API search is crawled from Twitter after being matched with APIs crawled from PW. Generally, the process of the algorithm could be divided into two parts. The first part is utilizing the name and description information of lists to compute the semantic similarity, including three steps: pre-processing, LSI and similarity calculation. After the data pre-processing like tokenization, stop words removal, stemming, etc., the features extracted from lists and the given query could be mapped into a conceptual space through LSI.

The returned relevant APIs are ranked according to cosine similarity, thus the semantics based rank score of an API and a given query is defined as:

$$score_s(q, API_i) = sim(q, d_i) = \frac{\sum_j w_{q,j} w_{i,j}}{\sqrt{\sum_j w_{q,j}^2} \sqrt{\sum_j w_{i,j}^2}}, \quad (1)$$

where \mathbf{w} denotes term vector of an API or query in latent semantic space. Another part is about popularity calculation for each API based on its list number, which reflects how popular an API is among Twitter users to a certain extent. The popularity based rank score of an API is define as:

$$score_p(API_i) = \frac{\log_{10} N - \log_{10} min}{\log_{10} max - \log_{10} min} \quad (2)$$

where N represents the number of lists including the i th API, min and max respectively denote the minimal and maximal numbers of APIs' lists.

The final rank score is decided by the integration of semantic similarity and API's popularity, since the two measures synthetically satisfy the demands of users. Thus, combining the last two equations, we define the final rank score as:

$$score(q, API_i) = \lambda score_s(q, API_i) + (1 - \lambda) score_p(API_i), \lambda \in [0, 1] \quad (3)$$

where λ is a weight to balance the importance of semantic similarity and popularity.

3 User Interface

Our crowdsourcing based API search engine is online available, users can use it to search APIs by visiting <http://zjums.com/projects/case/index.php>. The overall pages in the search platform are designed based on HTML5. The first page offers the crowdsourcing based API search function of the platform, and the next pages show some information and analysis about APIs, Twitter lists, and the demo video².

Figure 3(a) depicts the search interface and 52 sample queries from 12 categories are offered here. Users can click any sample query or input a query manually for searching the appropriate APIs. Figure 3(b) shows the search result page while user chooses a sample query *holiday*. It can be easily found that each search result entity mainly includes four parts: (1) API name; (2) API description; (3) the category API belongs to; (4) the lists number including API which shows API's popularity. Moreover, the first icon beside the API name links to the API page in ProgrammableWeb, which offers more detailed information about the API. When click the second icon, it will show the page of Twitter lists including the API, helping users to better understand the crowdsourcing knowledge about API.

² The demo video also can be found at <https://youtu.be/D6xTzFkAXjQ>.

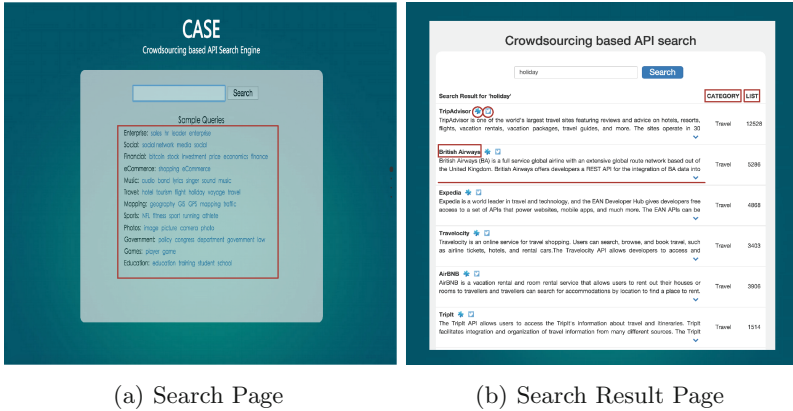


Fig. 3. User interface of CASE

4 Conclusion and Outlook

This paper demonstrates a crowdsourcing based API search platform named CASE. CASE applies LSI model to calculate semantic similarity based on Twitter lists information, and integrates it with API popularity to infer the final search result. Therefore, the search platform offers semantics matched and popularity satisfied results for a user query.

To enrich the function of the API search platform, it is considered to assign tags for each API which can be used to find APIs with similar function, and recommend APIs for a specific mashup in our future work.

Acknowledgment. This research was partially supported by the National Technology Support Program under grant of 2011BAH16B04, the National Natural Science Foundation of China under grant of 61173176, Science and Technology Program of Zhejiang Province under grant of 2013C01073, National High-Tech Research and Development Plan of China under Grant No. 2013AA01A604.

References

1. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. *JASIS* **41**(6), 391–407 (1990)
2. Ghosh, S., Sharma, N., Benevenuto, F., Ganguly, N., Gummadi, K.: Cognos: crowdsourcing search for topic experts in microblogs. In: *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 575–590. ACM (2012)
3. Kallen, N.: *Twitter blog: Soon to launch: Lists* (2009)
4. Lamere, P.: Social tagging and music information retrieval. *J. New Music Res.* **37**(2), 101–114 (2008)

WSTP: Web Services Tagging Platform

Sana Sellami^(✉) and Hanane Becha

Aix-Marseille Université, CNRS, LSIS UMR 7296, 13397 Marseille, France
sana.sellami@lsis.org, hbecha@gmail.com

Abstract. Recently tagging has been employed to improve the performance of service discovery. Two main challenges have to be addressed when tags are used in Web service discovery: tag relevancy and tag sense disambiguation. In this paper, we present our Web service tagging platform that addresses these problems and allows a semantic search of tagged Web services.

Keywords: Web service discovery · Tag · Semantic

1 Introduction

Tagging is the process of describing a resource by assigning textual keywords (tags) to it as a classification mechanism. The use of tags predates computers. In computer based systems, traditionally, tags were assigned by Web page developers and online databases publishers to help users find content. With the growth of social networking and multimedia sharing, user-contributed tags have gained wide popularity. Recently, tagging was employed to improve the performance of service discovery [1, 2]. There are two main challenges that have to be addressed when tags are used in Web service discovery: *tag relevancy* and *tag sense disambiguation*. First, to address the tag relevancy issue, we recommend the addition of *three* parameters to each tag: *score*, *popularity*, and *occurrence*. A score is assigned to each tag to denote the relevance of that tag from the user's perspective. Popularity denotes the relevance of a given tag according to the user's expertise (The user's expertise is assessed based on a simple three question quiz.). More weight is given for the most experienced users' tags. Occurrence is the number of times that a given tag was added to the same service. Second, to address the tag sense disambiguation, we use the WordNet dictionary to take into account the synonyms of the tags in the service search.

Our collaborative tagging Web Services Platform (WSTP) is unlike existing tagging systems (e.g., Titan¹, APIs.io²); since the Web services discovery is based on a search using tags. The WSTP returns the most relevant services according to the tags defined in the user's request and the tags of the available services. The remainder of the paper is organized as follows. Section 2 presents the architecture of the platform. Section 3, explains the details of the planned platform demonstration. Section 4 provides concluding thoughts.

¹ <http://ccnt.zju.edu.cn:8080/>.

² <http://apis.io/>.

2 System Overview

In this section, we provide an overview of our recommended platform (Fig. 1) for tagging Web services called WSTP (Web Services Tagging Platform).

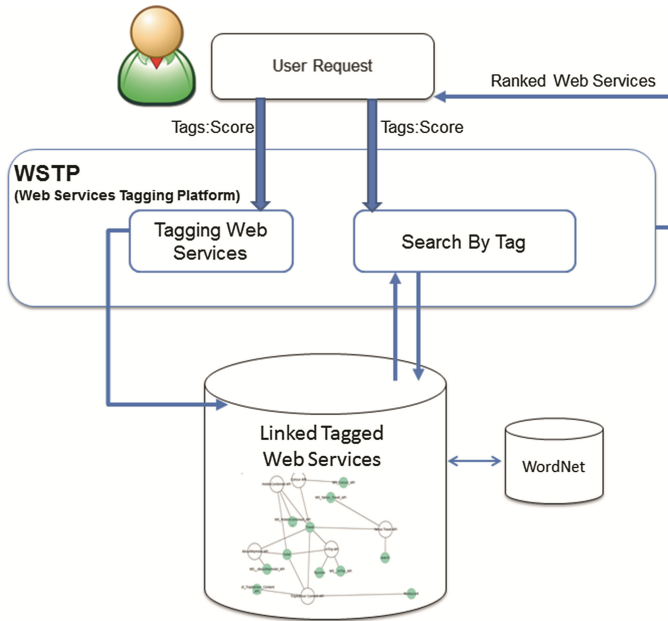


Fig. 1. WSTP Architecture

As illustrated in Fig. 1, users are empowered to perform two actions: assigning new tags to the available services and performing a tag-based search request. Each service has at least one tag which is by default the service name. The user-contributed tags are used to create linked services based on tag similarities and domains. Users can rely on service descriptions (if available) and/or on their expertise in web services to tag services manually.

2.1 Web Services Tagging Process

Users can improve the tag-based service discovery mechanism by adding new tags to each web service that they have invoked. To avoid malicious tags, users are required to register in the system before actively taking part in the tagging and searching of Web services. Even though one can argue that there are no wrong tags, it is logical that some tags could be more relevant than others with respect to a given service. To address the tag relevance problem, users are requested to add a score value to each tag that they add (see Definition 1).

Definition 1 (Tag Definition): A tag denoted as Tag (S) is a couple of name and a score: $\text{Tag}(S) = (\text{name}_{\text{Tag}}, \text{score}_{\text{Tag}})$ Where name_{Tag} and $\text{score}_{\text{Tag}}$ are respectively the name and the score of the tag where $\text{score}_{\text{Tag}} \in [0, 100]$.

2.2 Semantic Search of Linked Web Services

The semantic search process locates the most relevant services according to the user's request by performing a similarity search function. This search function takes a set of tags and their scores relevance as input to calculate the similarity between tags and their scores (see below Definition 2). During the search by tag, sense disambiguation techniques based on the WordNet³ lexical thesaurus are applied. In WordNet each term (tag) is a synset and it has a set of synonyms.

Definition 2 (Semantic Tag Similarity): The tag similarity denoted as $\text{Sim}_{\text{Tag}}(R, S)$, where R is a user request and S a service, is defined as follow:

$$\text{Sim}_{\text{Tag}}(R, S) = \text{Sim}_{\text{score}}(\text{TagR}, \text{TagS}) * W_{\text{score}} + \text{Sim}_{\text{sem}}(\text{TagR}, \text{TagS}) * W_{\text{sem}} \in [0, 1]$$

Where W_{score} and W_{sem} are weight set of the tag score and the semantic similarity, respectively, $W_{\text{score}} + W_{\text{sem}} = 1$ and:

$$\text{Sim}_{\text{score}}(\text{TagR}, \text{TagS}) = \frac{\sum |\min(\text{Score}(|\text{TagR} \cap \text{TagSi}|))|}{\sum \text{Score}(\text{TagR})}$$

is the tag similarity between the user's score request and services.

$$\text{Sim}_{\text{sem}}(\text{TagR}, \text{TagS}) = \frac{|\text{Tag}(R) \cap \text{Tag}(S)|}{|\text{Tag}(R)|}$$

is the semantic similarity between tags.

3 Demonstration

The WSTP system has been implemented in Java and it does interact with the MySQL database. The web pages are in JSP⁴. We rely on WordNet version 2.1. WSTP contains a collection⁵ of 151 Web services from different categories including Stock, Tourism, Weather, Telecommunication, Economy and Finance. The demonstration of WSTP platform shows the following features of our platform:

(1) Searching Web services by name, categories, tags or by browsing the linked services graph; (2) Adding Web services and Tags. This functionality requires user authentication; and (3) Searching Web services by setting a tag list with scores; A Web

³ <https://wordnet.princeton.edu>.

⁴ JavaServer Pages.

⁵ <http://www.zjujason.com/data.html>.

services tag and search example is demonstrated in our video which is available at: <http://www.lsis.org/sellamis/Projects.html#WSTP>.

We considered a set of four services in the DEMO category as illustrated in Table 1:

Table 1. Example of web services


Web services	Tags
WS1	sound:60 music:10 video:60
WS2	audio:70 social:40
WS3	dance:40 party:60 picture:80
WS4	music:80 sound:50 social:30 image:40

Consider a user request **R = music:80 sound:50 social:30 image:40 picture:50**.

The semantic search functionality compares the tags of R with the tags of services to find the most relevant services. Then, in the first iteration, we compared the occurrence of the tags and the name similarities based on the WordNet thesaurus. For each tag, we obtained a set of synonyms as described in Table 2. We retrieved the max (score) of each of the similar tags. All the tags that are synonyms were reduced to one tag with the most important, i.e. highest, score and we merged the rest of the tags. The result of this transformation is described in Table 3.

Table 2. Synonym results

Scores	Tags	Synonyms
40	image	image, picture, icon, ikon
50	picture	picture, image, icon, ikon
50	audio	audio, sound



Scores	Tags	Synonyms
50	picture	picture, image, icon, ikon
50	audio	audio, sound

Table 3. Web services similarities and ranking

Web services	Tags	Similarity	Ranking
WS1	sound:60 music:10 video:60	0.39	3
WS2	audio:70 social:40	0.44	2
WS3	dance:40 party:60 picture:80	0.24	4
WS4	music:80 sound:50 social:30 image:40	0.96	1

Based on the synonym transformations, we applied a semantic tag similarity function (see Definition 2) and obtained the following services as described in Table 3.

4 Future Work

We plan to enhance the semantic services search by querying a LOD (Linked Open Data) like DBpedia⁶ and interacting with Programmable Web⁷ for retrieving information on Web APIs from the repository. With this enhancement we will be able to provide a platform for linked services based not only on tag similarities but also on mashups, or category links.

References

1. Chen, L., Wu, J., Zheng, Z., Lyu, M.R., Wu, Z.: Modeling and exploiting tag relevance for Web service mining. *Knowl. Inf. Syst.* **39**, 153–173 (2014)
2. Fang, L., Wang, L., Li, M., Zhao, J., Zou, Y., Shao, L.: Towards automatic tagging for web services. In: 2012 IEEE 19th International Conference on Web Services, pp. 528–535 (2012)

⁶ <http://wiki.dbpedia.org/>.

⁷ <http://www.programmableweb.com/>.

Personalized Messaging Engine: The Next Step in Employee Engagement

Varun Sharma^(✉), Abhishek Tripathi, Saurabh Srivastava, Aditya Hegde,
and Koustuv Dasgupta

Xerox Research Centre India, Bangalore, India
{varun.sharma2,abhishek.tripathi3,saurabh.srivastava,
aditya.hegde,koustuv.dasgupta}@xerox.com

Abstract. Employers today are struggling to engage positively with their employees to reduce attrition and improve productivity. There are solutions in the market which are trying to solve the problem but they suffer from two critical issues. Firstly, the scope of the existing solutions is too narrow to capture each and every interaction happening within the company. Secondly, their learning from the employee behaviour is either non-existent or minimal at best. Personalized Messaging Engine (PME) is an attempt to provide end-to-end system to organizations for effective employee engagement. PME uses SOA principles to connect to each and every system through which employees engage with their employers. It uses the data aggregated from multiple systems to provide a hyper-personalized and dynamic experience to each employee. With the help of APIs, multiple systems can push data to PME and it then processes the data to send relevant pre-configured messages to the employees in the domain of Health, Wealth and Career. Additionally, PME uses several factors to prioritize messages for each and every employee. It uses a state-of-the-art learning engine to combine Subject Matter Experts opinion, Client Strategy, User Experiences and behaviour to find the messages which are most effective for the employees.

Keywords: Service oriented architecture · Recommendation engine · Personalized messaging · Employee engagement

1 Introduction

Employee engagement is crucial to a healthy relationship between an organization and its employees. According to [2], companies that do a better job at engaging employees do outperform their competitions. Companies today are generating data at a break neck pace at every touch point between employees and employers. There is no solution at this stage which captures all that data, analyses it and uses the analysis to improve the engagement levels between employee and employer. PME combines SOA principles and data analytics to create a solution which can fill the aforementioned gap. Given the monolithic nature of most of the applications used within a company, connecting all of them together

is a challenging task. PME makes use of several Data-as-a-Service applications to collect data from such applications.

In order to inspire employees to read and take action on messages, it is important not only to prioritize and present highly relevant messages to employees, but also to personalize prioritization by taking into account employees feedback (e.g., ratings based on usefulness of messages, actions taken on delivered messages etc.). In addition to employees preference for message prioritization, an organization will also have its own preferences for message prioritization. Thus, it is important to combine both organizations and employees preferences for the prioritization.

We propose using Collective Matrix Factorization (CMF) [1] to jointly model preferences from employees and employer along with other related data sets such as employee demography in terms of health, wealth, career and personal information. The CMF output can be used to (i) get a joint prioritization of messages, (ii) predict buying behavior (i.e. likelihood to act upon the messages) of employees for new messages and (iii) predict relevant messages for a new employee.

2 System Architecture

PME is a combination of several services talking to each other via APIs. In addition to standard ETL processes to gather data, data-as-a-service is used to aggregate data in multiple scenarios. At the core of PME are the Messaging Engine and Learning Engine which evaluate the eligibility of a message for an employee and also prioritizes the message for her. Using REST APIs, various channels such as HR Web Portals, Mobile Applications and Call Centre communicate with PME. Additionally, PME talks to an administrative console (again, via REST APIs) which allows employers to configure their HR strategies and messages.

2.1 Overview

Figure 1 depicts the PME system. The following two are the most important components of PME: (i) Messaging Engine and (ii) Learning Engine.

Messaging Engine: The Messaging Engine allows administrators to configure employers strategy and add messages in the domain of Health, Wealth and Career. Additionally, administrators can define rules based on which PME will decide the applicability of the message for an employee.

Learning Engine: The most important module within PME is the learning engine based on CMF. The learning engine looks at multiple data points to come up with a relevance score of a message for a given employee. In addition to employee data, following are the inputs for CMF:

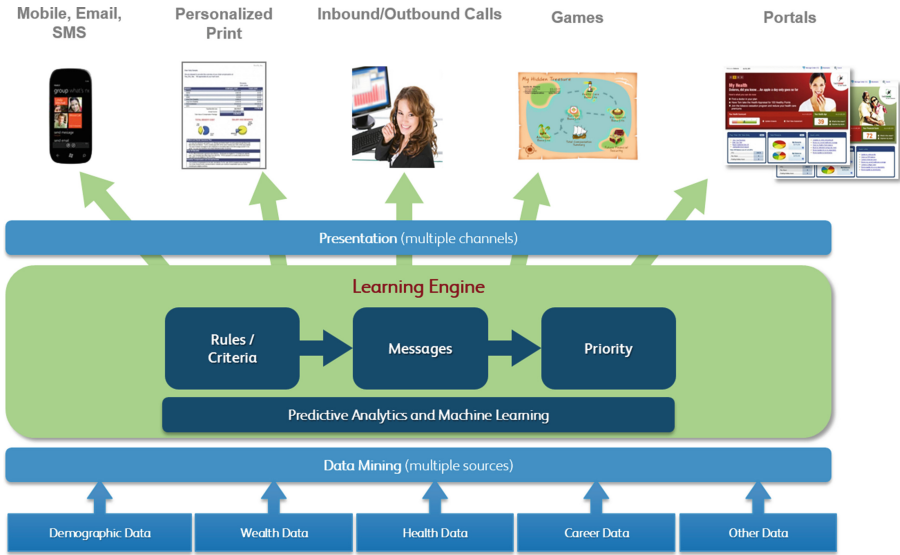


Fig. 1. Overview of PME system

1. **Risk.** A parameter defined by the Subject Matter Expert based on her view of the importance of the message.
2. **Initiative.** It captures the long term strategy of the employer and can be configured at multiple levels such as at a domain (Health, Wealth, Career) level, at a category (e.g. all preventive care messages) level and at an individual message level.
3. **Campaign.** It lets the employers control the priorities of the messages over a short duration based on annual events or seasonal events.
4. Feedback from the employees The feedback from the employees is captured at 3 different levels:
 - (a) **Level 1:** Employees are asked to rate a message and the rating is captured via a REST API call.
 - (b) **Level 2:** The interest of the employees is captured by storing whether they clicked on a hyperlink present in the message.
 - (c) **Level 3:** If an employee acts on a message and the same is reflected through a change in her profile, it is considered as a positive feedback.

2.2 SOA Based Integration

PME has been envisioned to be channel or client agnostic from the very beginning. We have achieved that by creating a multi-tenant web service which can be completely controlled through the REST APIs that it exposes. The system uses OAuth 2.0 to manage the authentication of the users accessing the system. It provides 3 different types of access controls

1. System Administrator are super users of the system who can access all its settings and make changes when needed.
2. Client Administrator can add or modify settings specific to a client of the PME system.
3. Client Users provide authentication settings which can be used by multiple channels (SMS, Web Portals, etc.) to pull relevant messages and send it to their users.

3 Features in PME

In the demonstration, we are going to talk about the capabilities of the PME through the following system:

1. A web portal which uses the PME REST APIs to show relevant messages to its users.
2. PME Admin Console which uses the REST APIs to configure PME settings for its clients.

During the demonstration, we will cover the following aspects of PME:

1. Use the admin console to onboard a new client.
2. Add/Modify messages in the client message library.
3. Run the Messaging Engine to compute the eligibility of the employees as per the criteria defined in the messages.
4. Demonstrate the ranking of messages for each employee after the Messaging Engine has finished its computation.
5. Demonstrate the capability of PME to capture feedback at multiple levels.
6. Demonstrate the capability of learning engine by using user feedback to re-prioritize the messages for each employee.

4 Conclusion

Personalized Messaging Engine is the first major step in using a holistic view of the activities in the company to create a more productive and engaged workforce. By using SOA techniques to combine disjoint systems to create that holistic view, PME analyses vast amount of data to provide the hyper-personalized experience required to keep the workforce happy and attrition rate low.

References

1. Klami, A., Bouchard, G., Tripathi, A.: Group-sparse embeddings in collective matrix factorization. In: ICLR 2014 (2014)
2. Crim, D., Gerard, H.S.: What engages employees the most or, the ten Cs of employee engagement. *Ivey Bus. J.* **70**, 1–5 (2006)

Offering Context-Aware Personalised Services for Mobile Users

Marie-Christine Fauvet¹(✉), Sanjay Kamath¹,
Isaac-Bernardo Caicedo-Castro², Pathathai Na-Lumpoon³,
Ahmed Lbath¹, and Lorraine Goeuriot¹

¹ LIG (MRIM), University of Grenoble Alpes, 38000 Grenoble, France

Marie-Christine.Fauvet@imag.fr

² University of Córdoba, Córdoba, Colombia

³ University of Chang-Mai, Chiang Mai, Thailand

1 Introduction

This paper presents a system that provides mobile users, context-aware personalized services. Users might need any sort of services: information about the weather, places to visit, accommodation booking, etc. The eTourism system is based on a semantic composition of recommended services into a composite service. A typical use case is as following:

Alice is an American tourist visiting Paris in France. At 4 PM, she wants to book a table at the finest restaurant in the city, and the direction to get there. So, she picks up her smartphone and accesses the system whose architecture is discussed in the next section (see Sect. 2), and issues the query: *I want to book a table for 2 people at the finest restaurant in the city, and I need the directions to the restaurant.*

The eTourism system will assist Alice by (1) analyzing her query; (2) capturing her context information; (3) processing the query, by identifying the right services, and composing them. (4) enacting the resulting composite services thus fulfilling Alice's needs.

Our main contribution is threefold: first, we have addressed service retrieval issues in order to build a module which discovers services queried by users; then, we have dealt with issues related with service operation automated composition to execute business processes, and finally, our system supports interactions with users to discover the missing parameters (not included in users' queries or their context information) in order to execute the business process. The expected result of the executable business process is able to fulfill the user's needs.

The demonstration introduced in this paper is accessible via the link <http://lig-membres.imag.fr/etourism/>.

2 System Architecture

The system we describe in this paper provides mobile users with context-aware personalized services according to their needs [5] when they are travelling. Figure 1

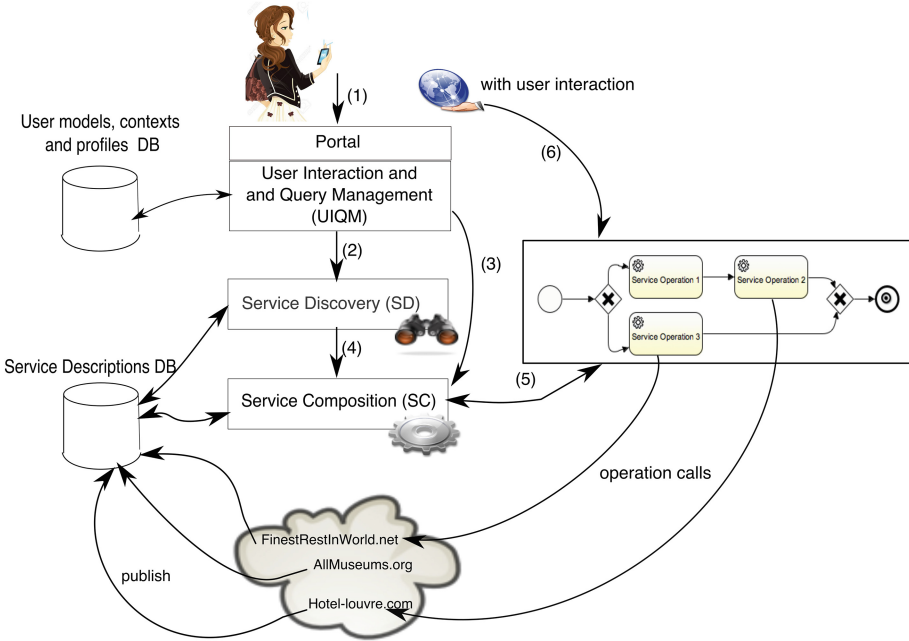


Fig. 1. A context-aware discovering system for mobile users

sketches the overall architecture of this system. The role of each module and the flow of information are detailed further in this section.

We consider the following definitions for context, profile and service concepts. A *context* includes spatial, temporal, physical, and environmental properties that could be collected by sensors embedded on the devices used to submit the queries. Such properties are for example: GPS location, timestamp, external temperature, screen size, etc. A *profile* captures users’ personal details, preferences and centres of interest.

We adopt the definition of service given by the W3C [6]: *A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL¹). Other systems interact with the web service in a manner prescribed by its description using SOAP² messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.*

The system whose architecture is sketched in Fig. 1, is accessible to registered users via a website on their mobile device (see Fig. 1, the data flow 1). This system is built on top of three components, each of which has been implemented as a service (the Portal is a web application).

¹ Web Service Description Language, <http://www.w3.org/TR/wSDL>.

² Simple Object Access Protocol.

User Interaction and Query Management (UIQM). This module aims at managing user connections and handling queries submitted by users. From data flow 1, this module receives the query and identification, and extracts from it the information needed to choose and compose service components. With data flow 2 the module sends queries to the *Service Discovery* module: if the query contains multiple requirements, it is split according to individual requirements and sent as single topic queries. With data flow 3, the users' requirements are sent to the *Service Composition* module. For more detailed description see [2].

Service Discovery (SD). Given the user's query received in the data flow 2, her profile and context, this module is responsible for retrieving services among a repository of service descriptions, that once composed can potentially meet the user's needs expressed in her query (for details see [1]). The retrieved services are then sent, following the data flow 4, to the next module.

Service Composition (SC). Eventually this module is in charge of automatically composing and executing services returned by the discovery phase. This module results in a BPMN³ model (see data flow 5) whose tasks refer to a service operation (for details see [3,4]). During the execution of the BPMN model some interactions with the user might be necessary to give value to parameters not found in the original query (see data flow 6).

3 Use Case

In order to illustrate the use case presented in Sect. 1, we will detail the process on the following query: *I want to book a table for 2 people at the finest restaurant in the city, and I need the directions to the restaurant.*

The *User Interaction and Query Management (UIQM)* module analyses the query to retrieve Alice's context and profile, along with the parameters needed for the requirements of the query.

The *UIQM* first captures Alice's context information: `coordinates="48.2167 N, 2.3332 E", current date="1/06/2014"`. Besides, the system has the following information about Alice's profile: `name="Alice", citizenship="USA", travelPurpose="tourism", gender="female"`.

The query considered in the use case has two requirements (restaurant booking, and directions), then the *UIQM* module splits this query into two subqueries and submits them to the *SD*. The first subquery is: *I want to book a table at the finest restaurant in the city*. The second subquery is *I need the directions to the restaurant*. The *SD* sends to the *SC* two ranked lists of candidate services, one for each subquery. From the list of candidate services that may fulfill the first subquery, the one with the highest rank contains the following operations:

1. *FindFinestRestaurant*: `cityName` → `name, address` which receives as a parameter the name of the city where Alice is looking for the finest restaurant. This operation returns the name and the address of the restaurant.
2. *BookRestaurant*: `name, people, phone` → `bookingConfirmation` which receives as parameters the restaurant name, the number of persons, and the user's telephone number. As a result, the operation returns a confirmation whether the table has been booked or not.

³ Business Process Management Notation (<http://www.bpmn.org>).

In the ranked list of candidate services that fulfill the second subquery, the highest ranked service contains the following operations:

1. **FromCoordinatesToCity**: *longitude, latitude* \rightarrow *cityName* which, given geographical coordinates, returns the name of the city whose location includes these coordinates.
2. **CoordinatesFromAddress**: *address* \rightarrow *longitude, latitude* which returns the geographical coordinates of the given address.
3. **GetDirection**: *origin, destination* \rightarrow *direction* which provides instructions on how to reach a destination. This operation receives two parameters, the coordinates of the starting point, and the coordinates of the destination.

The module *SC* takes as an input the operations of both services and compose them. The resulting composite is transformed into a BPMN model which in turn is executed on top of BPMN engine taken from the shelf. The resulting composite service fulfills both Alice's needs (i.e., booking a table at the finest restaurant of the city, and getting the directions to go there).

4 Conclusion and Further Work

In this work, we contribute with a context-aware system which provides mobile users with services, which are requested through queries in natural language.

For further work: (1) we will address service recommendation issues, aiming at provide services in push-mode, (2) we will tackle issues related with user privacy protection, (3) we will handle stateful services in business process model and (4) we will improve the user interaction by including into the system speech recognition to support spoken queries.

References

1. Caicedo-Castro, I.-B.: *S³niffer*: A text description-based service search system. Ph.D. Dissertation, University of Grenoble (2015)
2. Kamath, S.: Discovering context information and parameters from a natural language query. Masters MoSIG (2015). <https://hal.archives-ouvertes.fr/hal-01172050>
3. Na-Lumpoon, P.: Toward a framework for automated service composition and execution: E_tourism applications. Ph.D. Dissertation, University of Grenoble (2015)
4. Na-Lumpoon, P., Fauvet, M.-C., Lbath, A.: Toward a framework for automated service composition and execution. In: Proceedings of the 8th International Conference on Software, Knowledge, Information Management and Applications, Dhaka (2014)
5. Na-Lumpoon, P., Lei, M., Caicedo-Castro, I., Fauvet, M.-C., Lbath, A.: Context-aware service discovering system for nomad users. In: Proceedings of the 7th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), Chiang Mai, Thailand (2013)
6. W3C. Web services glossary (2004). www.w3c.org/TR/ws-gloss

Author Index

- Abe, Mari 461
Abu-Khzam, Faisal N. 345
Alatorre, Gabriel 139
Anyá, Obinna 139
- Barakat, Lina 53, 362
Barukh, Moshe Chai 218
Bazgan, Cristina 345
Becha, Hanane 486
Beheshti, Seyed-Mehdi-Reza 218
Benatallah, Boualem 218
Bermbach, David 154
Binotto, Alecio P.D. 324
Bouguettaya, Athman 333, 373
Bouloukakakis, Georgios 36
Bucchiarone, Antonio 383
- Caicedo-Castro, Isaac-Bernardo 495
Chander, Deepthi 444
Chang, Cheng 305
Chen, Chao 305
Chen, Junliang 87
Chen, Liang 482
Chen, Shiping 87
Chen, Xin 203
Chen, Zuoning 477
Chugh, Amandeep 427
Copil, Georgiana 105, 123
- Dasgupta, Gaargi Banerjee 412
Dasgupta, Koustuv 444, 491
De Sanctis, Martina 383
Di Cosmo, Roberto 397
Dijkman, Remco 237
Dong, Hai 333
Duan, Li 87
Dustdar, Schahram 105, 123
- Eiche, Antoine 397
Eshuis, Rik 285
Eswaran, Sharanya 427
- Fauvet, Marie-Christine 495
Fekete, Alan 316
Feng, Shichun 477
- Gajananan, Kugamoorthy 461
Gao, Bo 305
Georgantas, Nikolaos 36
Gerard, Scott N. 19
Ghari Neiat, Azadeh 373
Ghose, Aditya 412
Goeuriot, Lorraine 495
Gonzalez, Pavel 253
Gonzalez-Huerta, Javier 171
Grefen, Paul 237
Griesmayer, Andreas 253
Griffiths, Nathan 53
Guéhéneuc, Yann-Gaël 171
Gupta, Avantika 444
- Haddad, Joyce El 345
He, Ligang 305
Hegde, Aditya 491
Hoenisch, Philipp 316
Hull, Richard 285
- Issarny, Valérie 36
- Jagadeesh Chandra Bose, R.P. 444
Jain, Aditi 188
Jiang, Shun 461
- Kaes, Georg 269
Kalia, Anup K. 3, 19, 353
Kamath, Sanjay 495
Kattepur, Ajay 36
Keller, Alexander 139
Kuhlenkamp, Jörn 154
Kumar, Abhishek 427
- Langston, Bryan 139
Lbath, Ahmed 495
Le, Duc-Hung 105
Li, Kenli 305
Li, Keqin 305
Liang, Tingting 482
Lin, Pengxiang 477
Liu, Chuanchang 87
Liu, Chunhong 87

- Liu, Xumin 188
 Lomuscio, Alessio 253
 Luck, Michael 362
 Ludwig, Heiko 139

 Madden, John F. 3
 Mandagere, Nagapramod 139
 Mantripragada, Kiran 324
 Marconi, Annapaola 383
 Mauro, Jacopo 397
 Megahed, Aly 461
 Metzger, Andreas 71
 Miles, Simon 53, 362
 Mistry, Sajib 333
 Moha, Naouel 171
 Mohamed, Mohamed 139
 Moldovan, Daniel 105
 Mukherjee, Tridib 427
 Murukkannaiah, Pradeep K. 353

 Nakamura, Hiroaki 139
 Nakamura, Taiga 461
 Na-Lumpoon, Pathathai 495
 Netto, Marco A.S. 324
 Nguyen, Tien-Dung 105

 Palma, Francis 171
 Pistore, Marco 383
 Pohl, Klaus 71
 Pourmirza, Shaya 237

 Qin, A.K. 333

 Ramanath, Ajith 444
 Rinderle-Ma, Stefanie 269
 Rudolph, Kevin 154

 Schmieders, Eric 71
 Schulte, Stefan 316
 Sellami, Sana 486
 Sellis, Timos 373

 Shang, Yanlei 87
 Sharma, Varun 491
 Sikora, Florian 345
 Sindhgatta, Renuka 412
 Singh, Mridula 427
 Singh, Munindar P. 3, 19, 353
 Smith, Mark 461
 Srivastava, Saurabh 491
 Stamou, Katerina 139
 Sun, Yu-Jen John 218

 Taweel, Adel 362
 Taylor, Phillip 53
 Telang, Pankaj R. 3, 19
 Tizzei, Leonardo P. 324
 Traverso, Paolo 383
 Tremblay, Guy 171
 Tripathi, Abhishek 491
 Truong, Hong-Linh 105, 123

 Wadhwa, Rakshit 427
 Wang, Hongbing 203
 Weber, Ingo 316
 Wu, Hao 477
 Wu, Jian 482
 Wu, Qin 203

 Xie, Zhining 482

 Yadav, Kuldeep 427
 Yang, Wei 482
 Yi, Mengfei 285
 Yin, Jianwei 477
 Yu, Qi 188, 203

 Zacchiroli, Stefano 397
 Zavattaro, Gianluigi 397
 Zhao, Xinkui 477
 Zhi, Chen 477
 Zhu, Liming 316
 Zwolakowski, Jakub 397